Tennis ball tracking using Kalman filter

Gianvito Losapio
gvlosapio@gmail.com
ID 4803867

Abstract

Kalman filter, Kalman smoothing and Extended Kalman filter algorithms are briefly presented. An application of Kalman filtering and smoothing procedures is proposed to estimate the trajectories of tennis ball. In the provided IPtython notebook the Kalman filter implementation has been revisited with respect to the version proposed during lab activities, while the Kalman smoothing algorithm has been implemented from scratch. Real data coming from a private tennis club instrumented with a 3D stereoscopic system were used. Given an extremely simple physics linear model and sufficient accuracy of measurements, the Kalman smoothing solution was shown to be a trivial yet effective strategy to deal with relatively small temporal holes due to missing frames.

1 Introduction

The Kalman filter was originally proposed in (Kalman, [1]). It is one of the most celebrated and popular data fusion algorithms in the field of information processing. The most famous early use of the Kalman filter was in the Apollo navigation computer that took Neil Armstrong to the moon, and (most importantly) brought him back. Today, Kalman filters are at work in every satellite navigation device, every smart phone, and many computer games (Faragher, [2]).

In the present work I present an application of the Kalman filtering and smoothing algorithms to the tennis ball tracking problem. During last years, several methods were proposed in the literature, ranging from the use of particle filter to machine learning techniques. Today, the de facto standard for 3D tracking and line calling is represented by the Hawk-eye technology, whose preliminary results were presented in (Owens *et. al*, [3]) and now is available as a service provided by a dedicated company[1]. Not surprisingly, impact point detection in Hawk-eye systems is performed by combining interpolation methods with the use of Kalman filter.

In the following sections, the Kalman filter (sec. 2), the Kalman smoothing (sec. 3) and Extended Kalman filter (sec. 4) algorithms are presented. Details of the implementation, experiments and results are provided in section 5.

2 Kalman filter

In this section I briefly present the Kalman filter, trying to mix the presentations found in (Faragher, [2]) and (Murphy, [4]). The implementation is in the function `kalman_filter` of the notebook file.

The Kalman filter deals with Linear-Gaussian State Space Models (henceforth LG-SSM), that is to say evolution over time of linear dynamical systems (LDS) where every uncertainty is modeled through additive white Gaussian noise (AWGN).

A LG-SSM is composed of the following two main equations (where I assume that the system is stationary and described by a n-dimensional state vector x):

[1]https://www.hawkeyeinnovations.com/sports/tennis

1. *transition* or *process model*, describing how the state of the system evolves from time $t-1$ to time t:
$$x_t = \Phi\, x_{t-1} + B\, u_t + w_t \tag{1}$$
where:
    - x_t is the $n{\times}1$ state vector at time t
    - Φ is the $n{\times}n$ state transition matrix which applies the effect of each system state parameter at time $t-1$ on the system state at time t
    - x_{t-1} is the $n{\times}1$ state vector at time $t-1$
    - B is the $n{\times}n$ control input matrix which applies the effect of each control input parameter at time t on the state vector at time t (instantaneous system)
    - u_t is the $n{\times}1$ control input vector at time t
    - w_t the $n{\times}1$ vector containing the process noise terms for each parameter in the state vector. It is assumed to be drawn from a zero mean multivariate normal distribution with $n{\times}n$ covariance matrix Q, namely $w_t \sim \mathcal{N}(0, Q)$.
2. *observation* or *measurement model*, describing how a set of d measured parameters of the system we have access to for each time t relates to the system state:
$$z_t = H\, x_t + v_t \tag{2}$$
where:
    - z_t is the $d{\times}1$ vector of measurements available at time t
    - H is the $d{\times}n$ transformation matrix that maps the state vector parameters into the measurement domain
    - x_t is the $n{\times}1$ state vector at time t
    - v_t is the $d{\times}1$ vector containing the measurement noise terms for each term in the measurement vector. Like the process noise, it is assumed to be drawn from a zero mean multivariate normal distribution with $d{\times}d$ covariance matrix R, namely $v_t \sim \mathcal{N}(0, R)$.

It is worth noting that the LG-SSM model presented so far is represented by a set of parameters $\theta = \{\Phi, B, H, Q, R\}$ and any evolution of the process and the measurements from a time t_0 to a time t is determined by an initial state x_0 and a sequence of inputs $u_{1:t}$.

Assuming that the knowledge of the initial state is not deterministic and its uncertainty is again represented by a Gaussian pdf with $n{\times}n$ covariance matrix P_0, the LG-SSM model is fully specified for each time t by three Gaussian density functions:

$$\text{LG-SSM}(t_0 \to t) = \begin{cases} \text{Initial state: } x_0 \sim p(x_0) = \mathcal{N}(x_0, P_0) \\ \text{Process model (eq. 1): } x_t \sim p(x_t \mid x_0, u_{1:t}, \theta) = \mathcal{N}(\Phi\, x_{t-1} + B\, u_t, Q) \\ \text{Observation model (eq. 2): } z_t \sim p(z_t \mid x_0, u_{1:t}, \theta) = \mathcal{N}(H\, x_t, R) \end{cases}$$

The top picture in figure 1 provides an immediate interpretation of such probabilistic scenario. Given a fully specified LG-SSM, the Kalman filter is an algorithm which computes online (as the data streams in) its evolution over time by providing a prediction $\hat{x}_t$ of the true state vector x_t for each time t. Usually, x_t is referred as *hidden state* and $\hat{x}_t$ as *belief state*.

It is worth noting that this is defined as a *filtering* operation because it is aimed at reducing the uncertainty of the estimation by combining all the knowledge available in the LG-SSM. Intuitively, we could improve our prediction $\hat{x}_t$ by taking into account both our prior knowledge about the system and the measurements we have access to as the system evolves in real time.

But how to properly combine this knowledge? It is possible to show that the Kalman filter provide us the best possible solution to address the problem. Since everything is Gaussian in a LG-SSM, this is accomplished by exploiting a key property of the Gaussian function at this point: the product of two Gaussian functions is another Gaussian function. This is crucial since an endless number of Gaussian pdfs can be multiplied over time yet the resulting function does not increase in complexity or number of terms and the new pdf is fully represented again by a Gaussian function.

This is at the core of the elegant iterative procedure of the Kalman filter algorithm. Starting from the initial state x_0, fusing data to predict x_t for each time t corresponds to compute parameters of new

Gaussian pdfs:

$$\boxed{\begin{array}{lll} \text{LG-SSM}\,(t_0) & \leftarrow & x_0 \sim p(x_0) = \mathcal{N}(x_0, P_0) \\ \text{LG-SSM}\,(t_1) & \leftarrow & x_1 \sim p(x_1 \,|\, x_0, P_0, u_1, z_1, \theta) = \mathcal{N}(\mu_1, \Sigma_1) \\ \vdots & & \vdots \\ \text{LG-SSM}\,(t) & \leftarrow & x_t \sim p(x_t \,|\, \underbrace{x_{t-1}, P_{t-1}, u_t, z_t, \theta}_{\text{given information at time } t}) = \mathcal{N}(\underbrace{\mu_t, \Sigma_t}_{\text{target}}) \end{array}} \tag{3}$$

By letting

$$\hat{x}_t \leftarrow \mathbb{E}\big[\mathcal{N}(x_t \,|\, \mu_t, \Sigma_t)\big] = \mu_t \quad \forall t \tag{4}$$

the Kalman filter is the optimal estimator in the sense of being, from the set of all possible filters, the one which minimises the mean squared error (or equivalently maximises the likelihood) in the prediction of the state vector of a LG-SSM model.

The bottom picture in figure 1 clearly summarizes in a pictorial way this key intuition behind the Gaussian data fusion used in the Kalman filter.

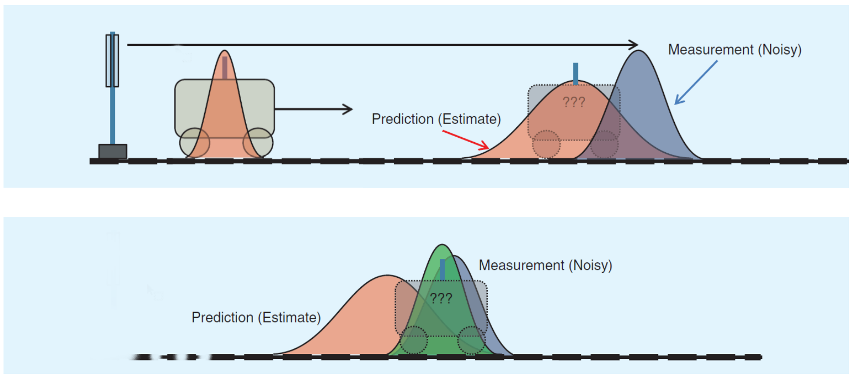


Figure 1: Visualization of probability density functions involved in a LG-SSM model for a simple one-dimensional tracking problem: a train moving along a railway line. (*Top*): The left red Gaussian distribution represents the pdf associated to the initial knowledge of the position of the train at time t_0. The overlapping Gaussian represent the pdf associated to the transition model (in red) and to the observation model (in blue) for a certain time t. (*Bottom*): For each time t the best guess about the position of the train is obtained by fusing the prediction coming from the transition model and the measurements coming from the noisy sensors. Credits: (Faragher, [2])

The iterative procedure (3) for each time $t > t_0$ is splitted into two different steps:

1. *Prediction step*: provides a temporary estimate of the vector state and its covariance at time t according to the process model

$$\begin{aligned} \hat{x}_{t\,|\,t-1} &= \Phi\, \hat{x}_{t-1} + B\, u_t \\ P_{t\,|\,t-1} &= \Phi\, P_{t-1}\, \Phi^{\mathrm{T}} + Q \end{aligned} \tag{5}$$

   Such quantities are to be intended as the mean and the covariance matrix of the normal distribution $\mathcal{N}(\hat{x}_{t\,|\,t-1}, P_{t\,|\,t-1})$ obtained by fusing the Gaussian pdfs associated to the process model and to the vector state estimate $\hat{x}_{t-1}$.

2. *Update step*: provides the final estimate of the vector state and its covariance at time t - which are the target parameters of the Gaussian $\mathcal{N}(\mu_t, \Sigma_t)$ - by combining the temporary estimates $\hat{x}_{t\,|\,t-1}, P_{t\,|\,t-1}$ with the new measurement at time t. Defining:
   - $r_t = z_t - H\, \hat{x}_{t\,|\,t-1}$: $d{\times}1$ vector describing the difference between our predicted observation and the actual measurement, called *residual* or *innovation*

- $S_t = H\, P_{t\,|t-1}\, H^{\mathrm{T}} + R$: $d{\times}d$ covariance matrix of the residual r_t
- $K_t = P_{t\,|t-1}\, H^{\mathrm{T}}\, S_t^{-1}$: $n{\times}d$ matrix containing the coefficients of the correction factor used to compute the Gaussian mean fusion, called *Kalman gain*
- I : $n{\times}n$ identity matrix

the computation is

$$\begin{aligned} \hat{x}_t &= \mu_t = \hat{x}_{t\,|\,t-1} + K_t\, r_t \\ P_t &= \Sigma_t = (I - K_t\, H)\, P_{t\,|t-1} \end{aligned} \tag{6}$$

It is woth noting that optimal estimate $\hat{x}_t$ of the state vector at time t, considered as the mean μ_t of the new fused Gaussian pdf, is the temporary mean provided by the prediction step $\hat{x}_{t\,|\,t-1}$ plus a term which represents the fusion with the measurements at time t and consists of a wighted contribution of the residual r_t with K_t being the weight.

Assuming for simplicity that H is an identity matrix, the resulting Kalman gain is $K_t = P_{t\,|t-1}\, S_t^{-1}$, highlighting that it can be interpreted as the ratio between the covariance of the temporary state vector obtained through the process model and the covariance of the measurement error. This observation leads to an informal definition of two main different regimes of the Kalman filter.

It is useful to recall that the covariance matrices P_0, Q, R describe "how much do we trust" of - respectively - the initial state, the process model and the measurements. Assuming for simplicity they are diagonal (uncorrelated noise), their initialization conditions the Kalman filter outcome as follows:

- Trace$(Q) \ll$ Trace(R): $|K_t|$ will be small and a little weight will be placed on the correction term. This means that the process model is trusted more than the measurements and it will have the largest contribution in the final estimate.

- Trace$(R) \ll$ Trace(Q): $|K_t|$ will be large and a lot of weight will be placed on the correction term. This means that the measurements are trusted more than the process model and they will have the largest contribution in the final estimate.

There are two dominant costs in the Kalman filter algorithm: the matrix inversion S_t^{-1} to compute the Kalman gain matrix, which takes $O(d^3)$, and the matrix multiplication $(I - K_t\, H)\, P_{t\,|t-1}$ to compute the updated covariance, which takes $O(n^3)$. In practice, more sophisticated implementations of the Kalman filter are used, for reasons of numerical stability, like the information filter or the square root filter (Murphy, [4]).

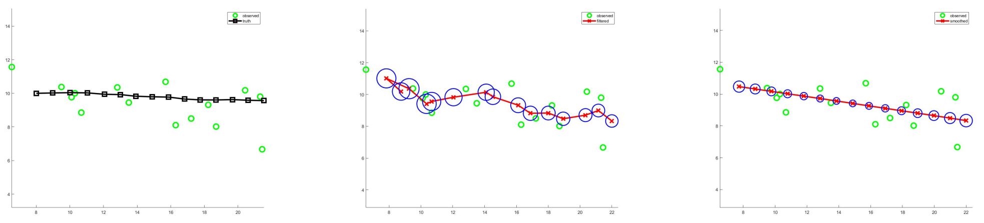
In the following sections two popular extensions of the Kalman filter are briefly presented.

3 Kalman smoothing algorithm

Kalman filter is formulated for online inference problems (i.e. tracking) for LG-SSM: as soon as a new measurement streams in, the prediction obtained according to the process model is optimally filtered out. However, if there are no online constraints, a different algorithm can be considered in order to achieve even better inference results.

As soon as the Kalman filtering procedure has been completed, it is possible to update the estimated belief states working backwards, sending information from the future back to the past and combining the two information sources. This operation significantly reduces the uncertainty of the estimation and provides better results.

This procedure is known as Kalman smoothing algorithm or RTS smoother, named after its inventors (Rauch et.al, [6]). The function `kalman_smoothing` of the notebook contains a simple implementation. An intutitive representation of the procedure is shown in fig. 2, then a brief insight of how it works follows.



(a) (b) (c)

Figure 2: Visualization of the difference between Kalman filtering and smoothing. (a) An object moves from left to right in the 2D plane and generates an observation at each time step. (b) Kalman filter results are shown, with blue circles being 95% confidence ellipses derived from the updated covariance. (c) Kalman smoothing results are shown. The confidence ellipsoids are smaller for the smoothed trajectory than for the filtered trajectory. The ellipsoids are larger at the beginning and end of the trajectory, since states near the boundary do not have as many useful neighbors from which to borrow information. Credits: (Murphy, [4], fig. 18.1, p. 634)

Assuming that $t = T$ is the time corresponding to the last belief state estimated with the Kalman filter, the backwards iterative algorithm proceeds as follows:

$$\boxed{\begin{array}{lll} \text{LG-SSM}\,(T) & \leftarrow & x_T \sim p(x_T) = \mathcal{N}(\mu_{T|T}, \Sigma_{T|T}) \\ & & \text{with } \mu_{T|T}, \Sigma_{T|T} \text{ initialized from the last Kalman filter output} \\ \vdots & & \vdots \\ \text{LG-SSM}\,(t) & \leftarrow & x_t \sim p(x_t \mid x_{t|t}, P_{t|t}, x_{t+1|T}, P_{t+1|T}, u_{t+1}, \theta) = \mathcal{N}(\underbrace{\mu_{t|T}, \Sigma_{t|T}}_{\text{target}}) \\ & & \text{with } x_{t+1|T}, P_{t+1|T}, u_{t+1} \text{ information sent backwards at time } t \\ \vdots & & \vdots \\ \text{LG-SSM}\,(t_0) & \leftarrow & x_0 \sim p(x_0 \mid x_0, P_0, x_{1|T}, P_{1|T}, u_1, \theta) = \mathcal{N}(\mu_{0|T}, \Sigma_{0|T}) \\ & & \text{with } x_0, P_0 \text{ initial state of the Kalman filter} \end{array}} \tag{7}$$

In particular, for each $t < T$, the step consists of the following equations:

$$\begin{aligned} x_t &= \mu_{t|T} = x_{t|t} + J_t(x_{t+1|T} - x_{t+1|t}) \\ P_t &= \Sigma_{t|T} = P_{t|t} + J_t(P_{t+1|T} - P_{t+1|t})J_T^{\mathrm{T}} \end{aligned} \tag{8}$$

where:

- $t|t$ refers to the Kalman filter estimation at time t
- $t + 1|T$ refers to the information sent backwards, starting from the last Kalman filter estimation and then considering the new smoothed values for next steps
- $t + 1|t$ refers to the prediction obtained at time t for time $t + 1$ according to the process model, equivalent to the prediction step of the Kalman filter
- $J_t = P_{t|t}\Phi^{\mathrm{T}}P_{t+1|t}^{-1}$ is the backwards Kalman gain matrix

It is woth noting that, analaogously to the Kalman filter algorithm, the smoothed estimate $\hat{x}_t$ of the state vector at time t, considered as the mean μ_t of a new fused Gaussian pdf, is the belief state provided by the Kalman filter $x_{t\,|\,t}$ plus a term which represents the fusion with the future data at time $t + 1$. Specifically, the fusion term consists of a wighted contribution of the residual between the new smoothed value $x_{t+1|T}$ and the corresponding prediction obtained with the process model $x_{t+1|t}$, with J_t being the weight.

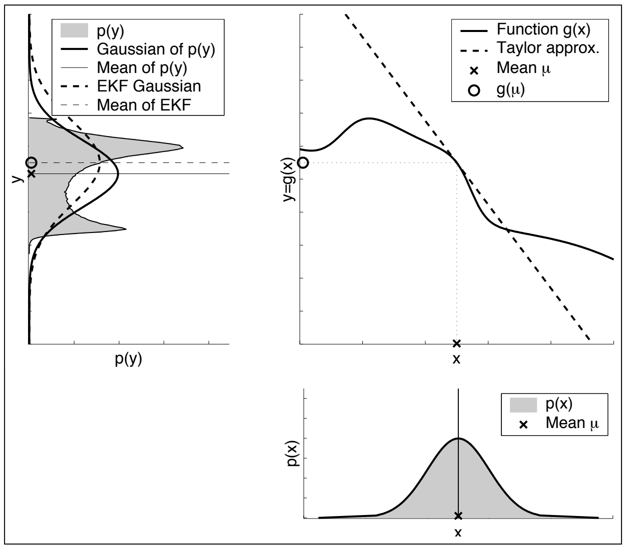


Figure 3: Visualization of the "mathematical trick" used in the EKF algorithm. The initial (previous) Gaussian state, shown on the bottom right, is passed through a non linear function, shown on the top right, to obtain the next belief state, producing a complex pdf shown in the shaded gray area in the top left corner. The dotted line is the EKF approximation to this: the Gaussian pdf is actually centered on the linearization of g and h about the previous state estimate using a first order Taylor series expansion. The solid line shows the best Gaussian approximation, being centered on the mean of the true complex distribution induced by the non linear transformation. Credit: (Murphy, [4], fig. 18.9, p. 651).

4 Extended Kalman filter

The extended Kalman filter (henceforth EKF) was formulated in order to deal with non-linear Gaussian space state models of the form:

$$\begin{aligned} \text{Process model:} \quad & x_t = g(x_{t-1}, u_t) + \mathcal{N}(0, Q_t) \\ \text{Observation model:} \quad & z_t = h(x_t) + \mathcal{N}(0, R_t) \end{aligned} \tag{9}$$

where

$$\begin{aligned} g(x_{t-1}, u_t) &= [g_1(x_{t-1}, u_t), \quad g_2(x_{t-1}, u_t), \quad \dots, \quad g_n(x_{t-1}, u_t)]^{\mathrm{T}} \\ h(x_t) &= [h_1(x_t), \quad h_2(x_t), \quad \dots, \quad h_d(x_t)]^{\mathrm{T}} \end{aligned} \tag{10}$$

are non linear but differentiable functions going from $\mathbb{R}^n$ to $\mathbb{R}^n$ and $\mathbb{R}^d$ respectively (following the dimensions introduced in par. 2).

Starting from an initial state $\mathcal{N}(x_0, P_0)$, a non-linear function induces a complex distribution which is not a Gaussian anymore. Nevertheless, the basic idea is to keep considering the new belief state represented by a Gaussian pdf centered in the value provided by the non linear functions, resorting to the following:

$$\text{Linearized G-SSM}(t_0 \to t) = \begin{cases} \text{Initial state: } x_0 \sim p(x_0) = \mathcal{N}(x_0, P_0) \\ \text{Process model: } x_t \sim p(x_t \mid x_0, u_{1:t}, \theta) \approx \mathcal{N}(g(x_{t-1}, u_t), Q) \\ \text{Observation model: } z_t \sim p(z_t \mid x_0, u_{1:t}, \theta) \approx \mathcal{N}(h(x_t), R) \end{cases}$$

Formally, this is possible thanks to a "mathematical trick" based on linearization of non linear transformation of Gaussian random variables, shown in fig. 3. The importance of such trick is to make the Gaussian property mentioned in section 2 still valid, allowing to recycle the standard Kalman filter equations as follows:

1. *Prediction step:*

$$\begin{aligned} x_{t\,|\,t-1} &= g(x_{t-1}, u_t) \\ P_{t\,|\,t-1} &= G_t\, P_{t-1} G_t^{\mathrm{T}} + Q \end{aligned} \tag{11}$$

   where G_t is the Jacobian matrix of g evaluated in x_{t-1}, u_t:

$$(G_t)_{ij} = \left.\frac{\partial g_i(x,u)}{\partial x_j}\right|_{(x_{t-1},u_t)}$$

2. *Update step:*

$$\begin{aligned} x_t &= x_{t\,|\,t-1} + K_t(z_t - h(x_{t\,|\,t-1})) \\ P_t &= (I - K_t\, H_t)P_{t\,|\,t-1} \end{aligned} \tag{12}$$

   where

   - H_t is the Jacobian matrix of h evaluated in $x_{t\,|\,t-1}$:

$$(H_t)_{ij} = \left.\frac{\partial h_i(x,u)}{\partial x_j}\right|_{x_{t\,|\,t-1}}$$

   - $K_t = P_{t\,|\,t-1}\, H_t^{\mathrm{T}}(H_t\, P_{t\,|\,t-1}\, H_t^{\mathrm{T}} + R)^{-1}$ is the Kalman gain matrix

The EKF works poorly when the initial covariance is large or when the function g is highly non linear near the current state estimate. In the case of well defined transition models, the EKF has been considered the de facto standard in the theory of nonlinear state estimation, navigation systems and GPS. Alternatively, unscented Kalman filtering and particle filtering algorithms are also largely used for approximate online inference of non linear state spaces (Murphy, [4]).

5 Experiments and Results

The data used for experiments are real data coming from a private tennis club instrumented with a 3D stereoscopic system made of 4 cameras arranged in two stereo pairs, one per each half court. They were kindly provided by the authors of the paper (Renò *et. al*, [7]), which is also the main reference for the entire experimental part of the project.

The original dataset comprised about 38000 raw frames @ 50 Hz (roughly 280 gigabytes of data for a total of 760 seconds $\simeq$ 13 mins) that represent a friendly match. 3D information of the ball were extracted through image processing techniques and through the application of projective transformations. The result is an intricated point cloud shown in Figure 4.

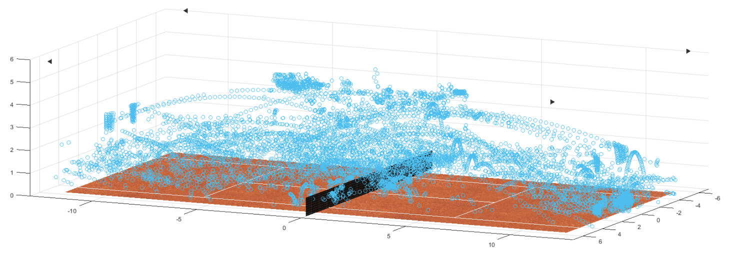


Figure 4: 3D point cloud data available for the experiments, extracted from $\sim$ 38000 raw frames @ 50 Hz. The black marker at the top corners show the position of the 4 cameras composing the stereoscopic system. Used with kind permission of Vito Renò and Ettore Stella.

From this initial set, only 583 points which have a corresponding ground truth were considered (manually annotated by highlighting the center of mass of the ball in the raw images and then applying the same projective transformations built in the stereoscopic system to extract 3D information).

Pieces of valid trajectories were identified using a clustering approach based on a customized temporal and spatial coherency (see the section Tracklets and Subtracklets of the IPython notebook file for further details). Each of the 45 final trajectories does not contain bounces nor strokes.

This step was necessary in order to adopt a simple linear physics model in the Kalman filter. The ball is treated as a point-mass adhering to just gravity along the Z-axis and subject to a constant velocity along the X and Y-axis (the origin is at the center of the tennis court). Besides simplicity, another reason behind the choice of such model is that there is no way to determine time at which the ball is hit and the input force or direction corresponding to this hit.

The vector state is composed of ball position and velocity. The resulting LG-SSM is the following:

1. Process model:

$$\begin{bmatrix} x_t \\ y_t \\ z_t \\ \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ z_{t-1} \\ \dot{x}_{t-1} \\ \dot{y}_{t-1} \\ \dot{z}_{t-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{2}(\Delta t)^2 \\ 0 \\ 0 \\ \Delta t \end{bmatrix} g + \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} \tag{13}$$

   where $\Delta t = 0.02\,s$ is the sampling period and $g = -9.80665\,m/s^2$ is the gravitational acceleration, $w \sim \mathcal{N}(0, Q)$

2. Observation model:

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ z_t \\ \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \end{bmatrix} + \begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix} \tag{14}$$

   where $v \sim \mathcal{N}(0, R)$

It should be noted that the ball is not always available in the three dimensional space, since the lack of information in just one camera makes the 3D estimation not possible. For this reason, support for missing measurements was included in the Kalman filter implementation.

The covariance matrices Q, R have been estimated from data. In particular, considering the total N observations of the residuals computed between points with matching frame number

$$r = \begin{bmatrix} r_x \\ r_y \\ r_z \\ r_{\dot{x}} \\ r_{\dot{y}} \\ r_{\dot{z}} \end{bmatrix} = \underbrace{\begin{bmatrix} x_{\text{ground truth}} - x_{\text{measured}} & \cdots & \cdots \\ y_{\text{ground truth}} - y_{\text{measured}} & \cdots & \cdots \\ z_{\text{ground truth}} - z_{\text{measured}} & \cdots & \cdots \\ \dot{x}_{\text{ground truth}} - \dot{x}_{\text{measured}} & \cdots & \cdots \\ \dot{y}_{\text{ground truth}} - \dot{y}_{\text{measured}} & \cdots & \cdots \\ \dot{z}_{\text{ground truth}} - \dot{z}_{\text{measured}} & \cdots & \cdots \end{bmatrix}}_{6 \times N} \tag{15}$$

the empirical covariance C has been considered:

$$C_{jk} = \frac{1}{N} \sum_{i=1}^{N} (r_{ji} - \bar{r}_j)(r_{ki} - \bar{r}_k) \tag{16}$$

Kalman filter and Kalman smoothing algorithms were tested through three different experiments corresponding to a different initialization of the covariance matrices Q, R:

- $R = C[:3, :3]$, $Q = C$: approximately same trust on the model and the measurements
- $R = C[:3, :3]$, $Q = 4 * C$: more trust on the measurements
- $R = 16 * C[:3, :3]$, $Q = C$: more trust on the model

In all the above cases, the initial state for each trajetory has been "borrowed" from the set of corresponding measurements. Consequently, the initial covariance has been set equal to Q.

The same metric proposed in the paper is used in order to quantify the accuracy of the algorithms. In particular, the figure of merit ϵ_i is considered for each estimated point $(\hat{x}_i, \hat{y}_i, \hat{z}_i)$ with respect to the corresponding ground truth (x_i, y_i, z_i):

$$\epsilon_i = \frac{1}{3}\Big[(x_i - \hat{x}_i) + (y_i - \hat{y}_i) + (z_i - \hat{z}_i)\Big] \tag{17}$$

The same metric is computed for the measurement points and the results are compared in order to evaluate the effectiveness of the Kalman algorithms. Results are provided in tabular form (table 1) and graphical format (fig. 6). In addition, a single 3D trajectory estimation is provided in fig. 5. Assuming that 0.1 meters of uncertainty is an acceptable threshold to evaluate performances, the Kalman smoothing algorithm achieves 95% of accuracy on real data in the best case.

Table 1: Point distribution with respect to intervals defined by increasing figure of merit ϵ_i

ϵ_{min}	ϵ_{max}	Est.	Meas.
-0.01	0.01	0.46	0.68
-0.02	0.02	0.66	0.77
-0.05	0.05	0.86	0.92
-0.10	0.10	0.95	0.97
-0.15	0.15	0.97	0.97
-0.20	0.20	0.97	0.97
-0.25	0.25	0.97	0.97
-0.30	0.30	0.97	0.97
-0.40	0.40	0.97	0.97
-0.50	0.50	0.97	0.97

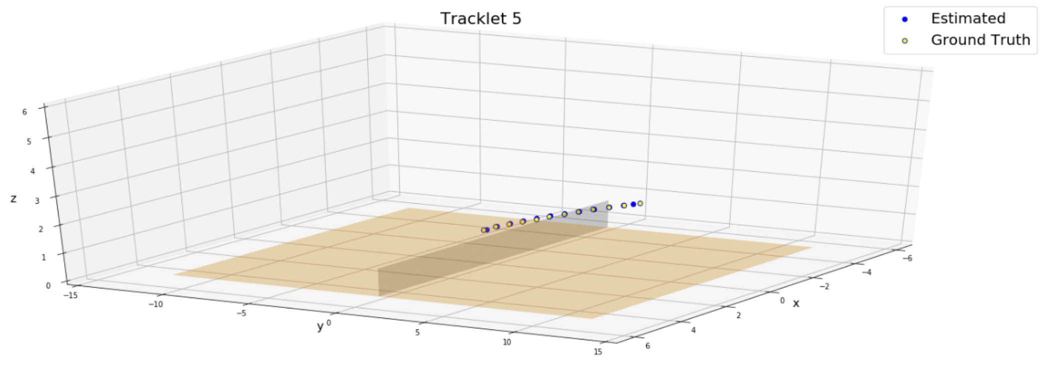


Figure 5: Qualitative result

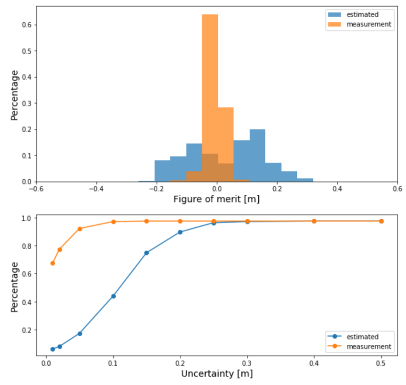
6 Conclusion

The best result has been achieved in the second experiment, when more trust was put on the measurements. The results are poorer in the other experiments as more and more importance is given to the model.

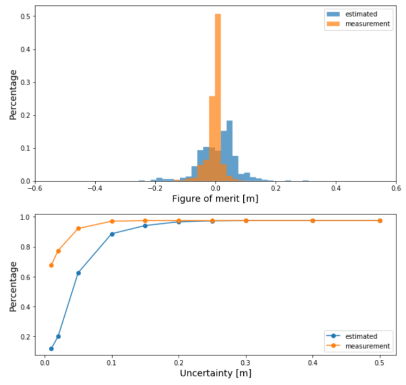
Such behaviour was expected since the physics model assumed for the filtering and smoothing procedures is extremely simple. As a consequence, the measurements are the best choice to provide a good estimation of the state. However, given sufficient accuracy of measurements, the Kalman smoothing solution was shown to be a trivial yet effective strategy to deal with relatively small temporal holes due to missing frames.

In general, the Kalman smoothing algorithm achieved better results with respect to the Kalman filter. This is due to the clear advantage of the offline processing not subject to online constraints, as explained in section 3.

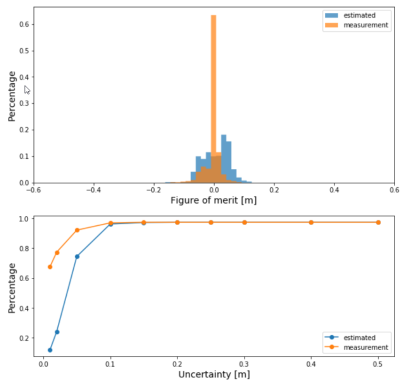
Clearly, more complex physics model should be taken into account in order to increase the accuracy of the tracker. In particular, taking into account air friction, estimation of input forces and directions, it should be possible to exploit the advantage of more advanced inference procedures for non linear models, like the Unscented Kalman filter or the particle filter (as proposed in the literature). It should be also possible to merge the mentioned tracking algorithms with the interpolation procedure proposed in the paper, following the strategy proposed by the Hawk-eye system [3].



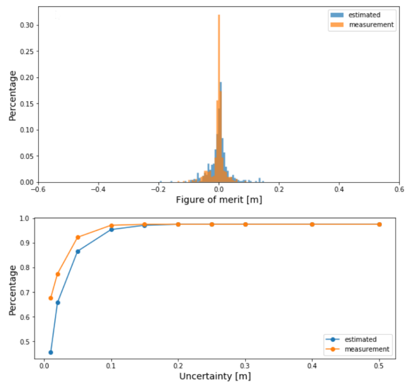
(a)



(b)



(c)



(d)

Figure 6: Results are provided through: histogram of figure of merit against percentage of points, cumulative percentage of points against increasing uncertainty (corresponding to the table 1). On the left side there are Kalman filter results, while Kalman smoothing results are on the right: **(a),(b)** first experiment $R = C[:3, :3]$, $Q = C$; **(c),(d)** second experiment $R = C[:3, :3]$, $Q = 4 * C$.

References

[1] Kalman R. E., *A new approach to linear filtering and prediction problems*, J. Basic Eng., vol. 82, no. 1, pp. 35–45, Mar. 1960.

[2] Faragher R., *Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation [Lecture Notes]*, in IEEE Signal Processing Magazine, vol. 29, no. 5, pp. 128-132, Sept. 2012.

[3] Owens N. et. al., "*Hawk-eye tennis system,*" in Visual Information Engineering, 2003. VIE 2003. International Conference on, July 2003, pp. 182–185.

[4] Murphy K., *Machine learning: a probabilstic perspective*, MIT Press, 2012

[5] Sorenson H.W., "*Least-squares estimation: from Gauss to Kalman*," in IEEE Spectrum, vol. 7, no. 7, pp. 63-68, July 1970.

[6] Rauch, H. E. *et. al.*, (1965). *Maximum likelihood estimates of linear dynamic systems.* AIAA Journal 3(8), 1445–1450.

[7] Renò V. *et. al.*, *Real-time tracking of a tennis ball by combining 3D data and domain knowledge*, 2016 1st International Conference on Technology and Innovation in Sports, Health and Wellbeing (TISHW), Vila Real, 2016, pp. 1-7

[8] Renò V. *et. al.*, *A technology platform for automatic high-level tennis game analysis, Computer Vision and Image Understanding*, Volume 159, 2017, Pages 164-175

[9] Fazio M. *et.al*, *Tennis Ball Tracking: 3D Trajectory Estimation using Smartphone Videos*, Final Project for EE367/CS448I: Computational Imaging and Display, Stanford University - Winter 2018, available online `https://stanford.edu/class/ee367/Winter2018/fazio_fisher_fujinami_ee367_win18_report.pdf`