Project Description

Francesco Sasso

I. Discrete-Time Kalman Filter Algorithm

The problem we are seeking to solve is the continual estimation of a set of parameters whose values change over time. Updating is achieved by combining a set of observations or measurements $m(t)$ which contain information about the signal of interest $x(t)$. The role of the estimator is to provide an estimate $\hat{x}(t+\tau)$ at some time $t+\tau$. In our case of study, the signal will be:

$$x(t) = \begin{bmatrix} p_1(t) \\ p_2(t) \\ v_1(t) \\ v_2(t) \end{bmatrix} \in \mathbb{R}^4,$$

where (p_1, p_2) represent the position of a point particle, while (v_1, v_2) represent its velocity. In the following, we will present the *discrete-time Kalman filter algorithm*, which will be used to solve the above mentioned problem.

Notation: Before going on, we introduce some useful ingredients:

$x_t \in \mathbb{R}^4$	system state vector at time t
$m_t \in \mathbb{R}^4$	observation vector at time t
$u_t \in \mathbb{R}^4$	control vector
$\hat{x}_{t\|s} \in \mathbb{R}^4$	estimation of x_t based on time s, with $s \leq t$
$P_{t\|s} \in \mathbb{R}^{4\times4}$	covariance matrix of $(x_t, \hat{x}_{t\|s})$ given the measurements $m_1, \ldots, m_s$, with $s \leq t$
$K_t \in \mathbb{R}^{4\times4}$	Kalman gain matrix at time t
$A_t \in \mathbb{R}^{4\times4}$	state transition matrix
$B_t \in \mathbb{R}^{4\times4}$	input transition matrix
$H_t \in \mathbb{R}^{4\times4}$	output transition matrix
$Q_t \in \mathbb{R}^{4\times4}$	process noise transition matrix
$R_t \in \mathbb{R}^{4\times4}$	measurement noise transition matrix

At this point, we are ready to describe the algorithm: it generates a sequence $\{(\hat{x}_{t|t}, P_{t|t})\}_{t\geq0}$ by means of three steps: an initialization step, a prediction step, and a correction step.

Initialization: Set the elements $(\hat{x}_{0|0}, P_{0|0})$.

Prediction: Predicts the state and variance at time $t+1$ dependent on information at time t:

$$\begin{aligned} \hat{x}_{t+1|t} &= A_t \hat{x}_{t|t} + B_t u_t \\ P_{t+1|t} &= A_t P_{t|t} A_t^\top + Q_t \end{aligned}$$

Correction: Corrects the state and variance using a combination of the predicted state and the observation m_{t+1}:

$$\begin{aligned} K_{t+1} &= P_{t+1|t} H_{t+1}^\top (H_t P_{t+1|t} H_t^\top + R_{t+1})^{-1} \\ \hat{x}_{t+1|t+1} &= \hat{x}_{t+1|t} + K_{t+1}(m_{t+1} - H_{t+1}\hat{x}_{t+1|t}) \\ P_{t+1|t+1} &= (I_4 - K_{t+1}H_{t+1})P_{t+1|t}, \end{aligned}$$

where I_4 is the identity matrix of order 4.

Francesco Sasso is with the Department of Engineering, Università del Salento, Lecce, Italy, francesco.sasso@unisalento.it.

After the correction step, the prediction step can be looped in order to obtain new prediction of the state at times $t+2,\dots$; however, these last are not based on new measurements of the state.

In the practical implementation of the just described algorithm, we used as initial elements:

$$\hat{x}_{0|0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad P_{0|0} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Moreover, we used the following matrices and control vector:

$$A_t = \begin{bmatrix} 1 & 0 & .2 & 0 \\ 0 & 1 & 0 & .2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; B_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; H_t = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \; Q_t = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & .1 & 0 \\ 0 & 0 & 0 & .1 \end{bmatrix}, \; R_t = \begin{bmatrix} .1 & 0 & 0 & 0 \\ 0 & .1 & 0 & 0 \\ 0 & 0 & .1 & 0 \\ 0 & 0 & 0 & .1 \end{bmatrix}, \; u_t = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

II. Project Structure

In this section we provide a summary of the project structure used to implement the Kalman filter algorithm. First of all, we used the following global variables:

- `exit_flag`: this is an integer used to control the exit of the program. Specifically, the program exit when the user press the esc key.
- `prediction_flag`: this is an integer used to enable or disable the prediction step's loops after the correction step.
- `trail`: this is a vector containing the history of the states estimated by the Kalman filter iterates, i.e., it contains $\hat{x}_{t-s|t-s},\dots,\hat{x}_{t|t}$, with $s \geq 0$ being the lenght of `trail`.
- `trail_length`: this is an integer which represents the lenght of `trail`. It can be updated by pressing the left key or the right key.
- `noise`: this is an integer which indicates the measurement noise. It can be updated by pressing the up key or the down key.
- `measurement`: this is a vector containing the last measurements of the position of the point particle (in our case, the mouse's cursor), i.e., it contains $m_{t-s},\dots,m_t$, with $s \geq 0$ fixed.
- `x`: this is a vector representing the estimated state of the point particle at time t, i.e., $\hat{x}_{t|t}$.
- `P`: this is a the covariance matrix at time t, i.e., $P_{t|t}$.
- `prediction`: this is a vector representing the prediction of the state at time $t+s$, with $s \geq 0$ fixed. It is obtained enabling the prediction step's loops after the correction step.

The above mentioned global variables are manipulated through the following tasks:

- $\tau_{\texttt{main}}$: this thread is responsible for the initialization of all the global variables;
- $\tau_{\texttt{keyboard}}$: this thread periodically listens if user press a key and reacts accordingly. Specifically, if user press:
  - key up/down: then the `noise` is increased/decreased;
  - key right/left: then the `trail_length` is increased/decreased;
  - key space: then the `prediction_flag` is enabled or disabled;
  - key esc: then the `exit_flag` is activated and the program ends.
- $\tau_{\texttt{kalman}}$: this thread periodically measures the position of the mouse's cursor subjected to the given `noise`, and updates the variable `measurement` accordingly. After that, it applies the prediction and the correction steps of the Kalman filter in order to update the `trail` and, consequently, the variables `x` and `P`. Moreover, if the `prediction_flag` is enabled, it also updates the `prediction`.
- $\tau_{\texttt{display}}$: this thread periodically displays the `measurement` and the `trail` vectors and, if `prediction_flag` is enabled, it also shows the `prediction` variable. Moreover, it shows the user interface and provide the instruction to increase/decrease the `noise` and the `trail_length`, and to enable/disable the `prediction_flag`.

A schema summarizing the just described project structure is depicted in Fig. 1.

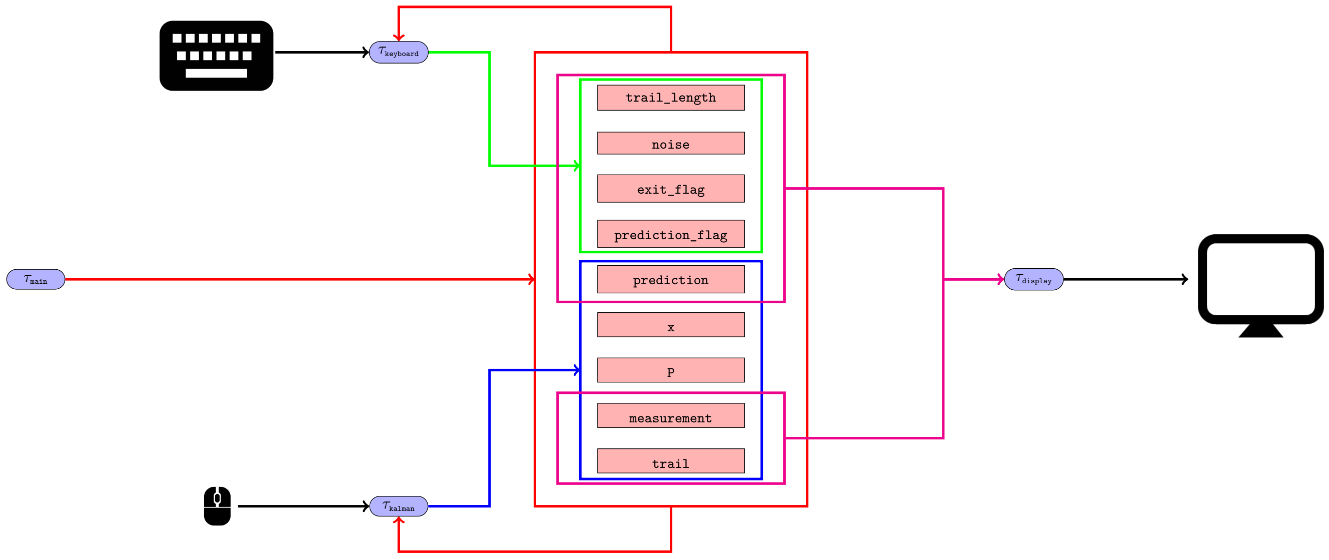

Fig. 1: Mind map representing the project structure.