A pedagogical approach to data fusion and kalman filter

José de Jesus Castillo-Zamora, Juan Pablo Aguilera-Alvarez, Juan-Antonio Escareno, Islam Boussaada, Juan Jose Martinez-Nolasco, Alonso Alejandro Jimenez-Garibay

A PEDAGOGICAL APPROACH TO DATA FUSION AND KALMAN FILTER

Castillo Zamora José de Jesús
L2S, Université Paris Sud-CNRS-CentraleSupelec, Université Paris Saclay
Institut Polytechnique des Sciences Avancées
jose.castillo@ipsa.fr

Aguilera Álvarez Juan Pablo
Tecnológico Nacional de México en Celaya
juan.aguilera@itcelaya.edu.mx

Escareño Castro Juan Antonio
XLIM UMR CNRS 7252, Université de Limoges
juan.escareno-castro@unilim.fr

Boussaada Islam
L2S, Université Paris Sud-CNRS-CentraleSupelec, Université Paris Saclay
Institut Polytechnique des Sciences Avancées
INRIA Saclay
islam.boussaada@centralesupelec.fr

Martínez Nolasco Juan José
Tecnológico Nacional de México en Celaya, Departamento de Ingeniería Mecatrónica
juan.martinez@itcelaya.edu.mx

Jiménez Garibay Alonso Alejandro
Tecnológico Nacional de México en Celaya, Departamento de Ingeniería Mecatrónica
alonso.jimenez@itcelaya.edu.mx

1. Introduction

The current paper is addressed to under graduated students interested in sensors and data fusion techniques. This work is written in an easy scientific language in order to be understood by the students. We propose the usage of the Kalman Filter as a first approach to data fusion as it is widely used in different research fields and applications as cited by [Sasiadek et al., 2000] and [Gao *et al.*, 2002].

In this matter, [Liggins et al., 2017] define Data fusion as the combination of the data from multiple sensors to achieve specific inferences that could not be achieved by the usage of one single sensor. Different techniques can be applied to carry out the aforementioned task, nevertheless Kalman Filtering is one of the most significant due to the robustness of the approach, the easy-reasoning-and-understanding of the

filtering principle, among other benefits and advantages discussed by [Gan et al., 2001] and [Sasiadek et al., 2000] who also mention several variations of this filter which improve the performance of the data fusion operation and/or estimation.

The literature offers a wide amount of papers and research concerning the application of Kalman Filtering techniques in data fusion, to mention some, [Sasiadek et al., 2000] used the Kalman Filter to improve the performance of autonomous robots to accomplish the navigation and trajectory-tracking tasks. [Zheng et al., 2019] compute the dynamic displacement of a given structure based on on-line multi-rate data fusion of high-sampling rate acceleration with time-varying bias and low-sampling rate displacement measurements. In addition, a linear Kalman Filter is implemented by [Castillo-Zamora et al., 2019] to meet the trajectory tracking specifications of a multi-link unmanned aerial system. More detailed examples can be find in the literature.

The methodology used in the paper consists in a full pedagogical discussion and representation about the Kalman filter for data fusion followed by the sensors characterization and the dynamics modelling of the systems to be studied in the examples. Simulations were carried out to illustrate the behavior of the systems with and without the Kalman filter. The conclusions obtained ae lastly exposed.

2. Methodology

To understand the Kalman filtering algorithm, let us consider our linear system to be described in a discrete domain by the expressions:

$$x_k = Ax_{k-1} + Bu_k + w_k \tag{1}$$

$$y_k = Hx_k + v_k \tag{2}$$

Equation 1 establishes that the current state of the system $x_k \in \mathbb{R}^n$ is given in some proportion ($A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$) by the previous state $x_{k-1} \in \mathbb{R}^n$ and the control input $u_k \in \mathbb{R}^m$. The term $w_k \in \mathbb{R}^n$ is called process noise and represents the influence of unmodeled phenomena. Equation 2 defines the measurement value $y_k \in \mathbb{R}^p$ as a linear combination of the current state (in some proportion $H \in \mathbb{R}^{p\times n}$) and the measurement noise $v_k \in \mathbb{R}^p$ which is normal distributed, i.e. Gaussian. One

should notice that the better the noise parameters are defined, the better estimates one gets.

Based on equation 1 and 2, the Kalman Filter equations are conceived such that these are divided in two sets: Time Update (prediction) and Measurement Update (correction). The first set is formed by the expressions:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \quad (3)$$

$$P_k^- = AP_{k-1}A^T + Q \quad (4)$$

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \quad (5)$$

While the correction set is defined by:

$$y_k = sensor\ measurement \quad (6)$$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - H\hat{x}_k^-) \quad (7)$$

$$P_k = (I - K_k H)P_k^- \quad (8)$$

Both sets are applied at each step or iteration in strict order, moreover, $I \in \mathbb{R}^{n\times n}$ stands for the identity matrix, $P_k \in \mathbb{R}^{n\times n}$ is the error covariance matrix and, $R \in \mathbb{R}^{p\times p}$ and $Q \in \mathbb{R}^{n\times n}$ are the covariance matrices of the environment noise and the process noise, respectively. Often, these ones result difficult to compute.

To start the process, we need to know the estimate of x and P at the beginning of the process, these values are called initial conditions and are represented as $\hat{x}_0$ and P_0, respectively.

The Filter explained above in equations 3-8 is often used for estimation tasks, in order to implement it as a data fusion technique some modifications and assumptions need to be taken into account.

Let us consider, a mechanical system whose states are the position and velocity, i.e. x and $\dot{x}$, and a pair of sensors at our disposal for knowing the position and acceleration, an ultrasonic sensor and an accelerometer, for instance.

The position captured by the ultrasonic sensor is written as $x_u \in \mathbb{R}$ that can be derivate w.r.t. time to obtain the velocity $\dot{x}_u \in \mathbb{R}$ and the acceleration $\ddot{x}_u \in \mathbb{R}$. By integrating the acceleration signal coming out from the accelerometer $\ddot{x}_a \in \mathbb{R}$, also the velocity $\dot{x}_a \in \mathbb{R}$ and the position $x_a \in \mathbb{R}$ can be known.

To estimate the states of the system with more accuracy, $\ddot{x}_a$ is used at the Prediction stage of the Kalman Filter and x_u is used in the correction stage, as shown next.

By simple kinematics the position and the velocity of the system can be obtained [Castillo-Zamora, 2016], i.e.

$$x = x_0 + \dot{x}t + \frac{1}{2}\ddot{x}t^2 \tag{9}$$

$$\dot{x} = \dot{x}_0 + \ddot{x}t \tag{10}$$

Where $x_0 \in \mathbb{R}$ and $\dot{x}_0 \in \mathbb{R}$ are the initial position and velocity of the system, $t \in \mathbb{R}^+$ stands for the time. The discrete representation of equations 9 and 10 is given as:

$$x_k = x_{k-1} + \dot{x}_{k-1}dt + \frac{1}{2}\ddot{x}_k dt^2 \tag{11}$$

$$\dot{x}_k = \dot{x}_{k-1} + \ddot{x}_k dt \tag{12}$$

Such that $dt \in \mathbb{R}^+$ is the sample time. The latter equations 11 and 12 can be comprised in a matrix form as follows

$$X_k = A_X X_{k-1} + B_X U_k \tag{13}$$

With

$$X_k = \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix};\ A_X = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix};\ B_X = \begin{bmatrix} dt^2/2 \\ dt \end{bmatrix};\ U_k = [\ddot{x}_k] \tag{14}$$

In our case, the vector U_k will contain $\ddot{x}_a$ measured at the instant k.The fact that we can observe the position by means of the ultrasonic sensor implies an observation equation as follows:

$$Y_k = H_X X_k\ ;\ H_X = [1 \quad 0] \tag{15}$$

The information provided by equations 13, 14 and 15 allows us to rewrite the Kalman filter equations 3-8 as shown below.

$$\hat{X}_k^- = A_X \hat{X}_{k-1} + B_X \ddot{x}_{a_k} \tag{16}$$

$$P_{X_k}^- = A_X P_{X_{k-1}} A_X^T + Q_X \tag{17}$$

$$K_{X_k} = P_{X_k}^{-} H_X^T \left(H_X P_{X_k}^{-} H_X^T + R_X \right)^{-1} \tag{18}$$

$$Y_k = x_{u_k} \tag{19}$$

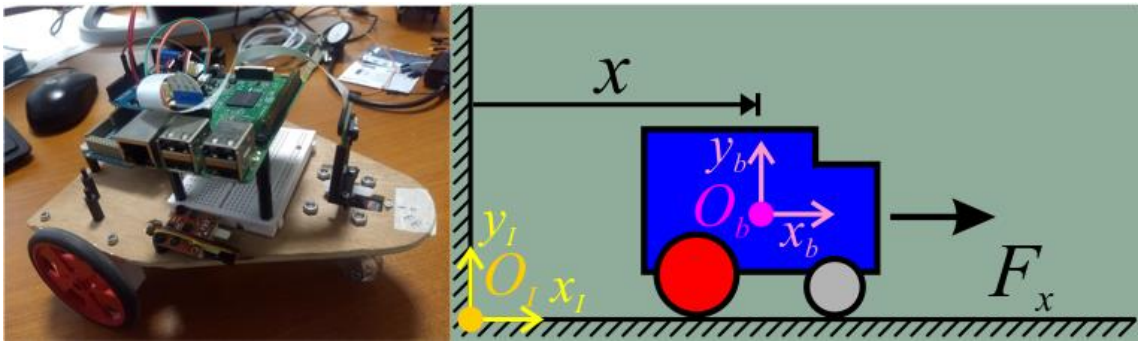
$$\hat{X}_k = \hat{X}_k^{-} + K_{X_k} \left(Y_k - H_X \hat{X}_k^{-} \right) \tag{20}$$

$$P_{X_k} = \left(I - K_{X_k} H_X \right) P_{X_k}^{-} \tag{21}$$

Notice that now, the signals of the ultrasonic sensor and the accelerometer take part of the Kalman filter equations also it is worth to mention that at the output of the filter, the vector $\hat{X}_k$ contains the states of the system computed from the measurement of both sensors thus it is used to close the control loop as explained later. Further and detailed information about the Kalman Filter theory is available in the references cited in this work.

2.1 Example 1: Mini car robot

The figure 1 shows the Mini car robot used in this example. The real platform can be observed at the left, meanwhile the free body diagram is depicted at the right. This vehicle features an Arduino Due board, a Raspberry Pi 3 connected to a camera, an IMU (Inertial Measurement Unit), an ultrasonic sensor and an Arduino Motor shield. The actuators of the system are two continuous servo motors. In the front of the robot a free-to-rotate wheel is placed to balance the system.



a) Real mini car robot *b) Mini car free body diagram*

Figure 1 The mini car robot.

For sake of simplicity, we are interested only in the translational linear movement of the car. Notice that in figure 1, two different frames have been defined: the inertial reference frame (O_I defined by the axis x_I and y_I) which is fixed to some point in the space and the body-fixed reference frame (O_b with the axis x_b and y_b) which is located at the center of gravity of the vehicle. The position of the vehicle is described by $x \in \mathbb{R}$ and measured as shown, additionally, the velocity and the acceleration of the car are represented as $\dot{x} \in \mathbb{R}$ and $\ddot{x} \in \mathbb{R}$, respectively defined by the first and second time derivatives.

In order to control the vehicle, it is important to have the mathematical description of it. The dynamic model can then be obtained by the Newton Euler formalism or by the Euler Lagrange formalism [Castillo-Zamora, 2016]. For the simplest case of our robot, by both procedures, the dynamics is found to be:

$$m_c \ddot{x} = F_x \tag{22}$$

Notice that the friction force between the wheels of the car and the floor has been omitted. $F_x \in \mathbb{R}$ represents the force applied over the vehicle along the x_I. The mass of the vehicle is expressed as $m_c \in \mathbb{R}^+$.

Let us assume that the force F_x can be easily manipulated, this allows us to define such force as the control input $u_x \in \mathbb{R}$, i.e.:

$$F_x = u_x \tag{23}$$

The main goal of the controller in this example is to make the vehicle go to a desired specific point in the space located at a distance $x_d \in \mathbb{R}$ in the x_I axis. To do so, we can apply different control techniques however, we are going to apply a PD controller [Castillo-Zamora et al., 2018]:

$$u_x = K_{p_x} e_x + K_{v_x} \dot{e}_x \tag{24}$$

where K_{p_x} and K_{v_x} (both $\in \mathbb{R}^+$) are the proportional and derivative gains of the controller, meanwhile $e_x = x_d - x \in \mathbb{R}$ is the position error and $\dot{e}_x = \dot{x}_d - \dot{x} = -\dot{x} \in \mathbb{R}$ is the velocity error. Notice that as the desired position is a fixed point in the space, i.e. it is a constant, its time derivative is always 0.

Based on equations 23 and 24, equation 22 can be rewritten such that:

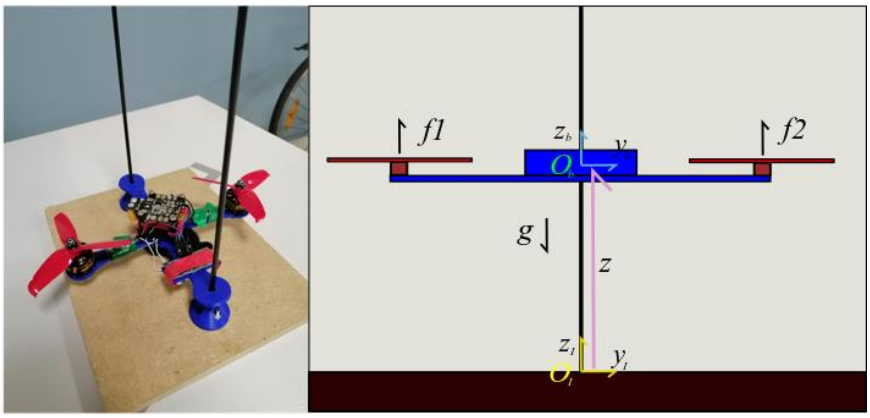
$$m_c\ddot{x} = K_{p_x}e_x - K_{v_x}\dot{x} \tag{25}$$

Equation 25 will be used at the simulation stage to implement the data fusion for merging the information of the sensors.

2.2 Example 2: Mini bi-rotor platform

Analogously to the analysis of the mini car robot, the modelling and control of the bi rotor shown in figure 2 are conceived in this subsection. The main features of the platform are the Arduino Nano board, a PDB or Power Distribution Board to power up the electronics as well as one IMU that measures the acceleration of the vehicle, an ultrasonic sensor to know the distance $z \in \mathbb{R}$ from the ground and two ESCs (Electronic Speed Controllers) to manipulate the brushless motors velocity.

We are interested only in the vertical linear movement of the vehicle. For sake of simplicity, the figure 2 depicts, at the right, a free body diagram of the system. Two different frames are established: the inertial reference frame (O_I defined by the axis y_I and z_I) which is a fixed point in the space and the body-fixed reference frame (O_b with the axis y_b and z_b) which is located at the center of gravity of the vehicle. The velocity and the acceleration of the vehicle are represented as $\dot{z} \in \mathbb{R}$ and$\ddot{z} \in \mathbb{R}$, respectively defined by the first and second time derivative.



a) Real Bi-rotor platform *b) Bi-rotor free body diagram*

Figure 2 The bi-rotor platform.

The mass of the aircraft is represented by $m_v \in \mathbb{R}^+$ and the gravity acceleration $g = 9.81\ m/s^2$ acts over the vehicle as depicted, defining the equation of motion, according to the Newton Euler formalism, to be:

$$m_v\ddot{z} + m_v g = f_1 + f_2 \tag{26}$$

In this case, the sum of the forces acts as the control, i.e.

$$f_1 + f_2 = u_z \tag{27}$$

Once more, the objective of the controller is to stabilize the vehicle at certain constant altitude $z_d \in \mathbb{R}$ above the ground. As in the previous example, a PD controller is used to accomplish such goal nevertheless we add the gravitational terms to the equation as follows:

$$u_z = K_{p_z} e_z - K_{v_z}\dot{z} + m_v g \tag{28}$$

with K_{p_z} and K_{v_z} (both $\in \mathbb{R}^+$) are the proportional and derivative gains of the controller, meanwhile $e_z = z_d - z \in \mathbb{R}$ is the position error. The definitions in equations 27 and 28 allow us to rewrite the dynamics of the closed loop system in equation 26 such that:

$$m_v\ddot{z} + m_v g = K_{p_z} e_z - K_{v_z}\dot{z} + m_v g \tag{26}$$

Further information about the dynamics of the vehicle and its control, as well as other models for pedagogical implementation can be found in [Castillo-Zamora et al., 2015], [Castillo-Zamora, 2016], [Escareño et al., 2017] and [Castillo-Zamora et al., 2018].

2.3 Sensors characterization

In order to study the proposed data fusion algorithm, the characterization and correct simulation of the sensors are a most. In this regard, the accelerometer can be simulated as a signal with an addition of some white noise (Gaussian) and a given offset, this because the accelerometer measures also the gravity acceleration and, depending on its attitude, the sensor gives back some acceleration value even when

the vehicle is not in motion [Awasthi et al. 2015] thus, the total acceleration measurement coming from the sensor is composed by the sum of three components: the real acceleration of the vehicle, the Bias acceleration (or Offset) and the Gaussian distributed noise. One should be warned that obtaining the velocity and position from the integration of the accelerometer information can produce significant errors due to the (almost) constant which becomes a linear function when integrated the first time and a quadratic function after the second integration.

The ultrasonic sensor, in simulation, has the advantage of not carrying a derivation accumulative error for both, velocity and acceleration, computations but presents the disadvantage of amplification of the velocity and accelerations as well as not considering the initial positions and velocities of the system as when using the accelerometer. In addition, the sensing process with this kind of sensors is slow compared with that of the accelerometer [Carullo et al., 2001]. For short, in Matlab/Simulink, this sensor signal will be composed by the real position of the vehicle and a Gaussian distributed noise with slower sampling time than that of the acceleration sensor previously described.

One should have in mind that most of the times, control simulations are carried out considering ideal elements (actuators and plants) that can provide the desired control command to the system which, in fact, does not happen in reality. Moreover, we must think of which kind of control signals we need, e.g. for translational mechanical systems the signal control is given (typically) in Newtons yet sometimes the actuators provide actions or signals that are related to displacements (meters or radians) and torques (Newton-meters) which implies that a conversion is needed, leading to more errors and less accuracy in the prediction of the response. In these examples, we consider only a saturation of the control input and assume that we can directly provide such command in the corresponding units.

3. Results

The proposed fusion data technique can be proved to be valid via numerical simulations, in order to do so, Matlab and Simulink provide the most appropriate tools and environment.

Table 1 Simulation parameters and conditions.

Simulation parameters			
Total simulation time		30 s	
Simulation sampling time (dt)		0.001 s	
Gravity acceleration (g)		9.81 m/s^2	
Sensors parameters			
Accelerometer variance		0.0001	
Mean Bias value		-0.97 m/s^2	
Bias variance		0.001	
Ultrasonic sensor variance		0.0001	
Ultrasonic sensor sampling time		$30 * dt$	
Mini car properties and parameters		**Bi-rotor properties and parameters**	
Mass	$1\ kg$	Mass	$0.5\ kg$
Initial position	$0\ m$	Initial position	$2\ m$
Initial Velocity	$0\ m/s$	Initial Velocity	$0\ m/s$
Proportional gain	1	Proportional gain	1
Derivative gain	1	Derivative gain	1
Desired position	$2\ m$	Desired position	$0.75\ m$

Before giving the detail information about the development of the simulation environment, let us introduced the Table 1 which contains all the parameters used in simulation as well as the corresponding value or magnitude.

For both systems, we defined the total simulation to be 30 s with a sampling time of 0.001 s in order to perform a fixed time step simulation. In the case of the Bi rotor platform the gravity acceleration is also established in the program according to Table 1. In addition, we assumed that the accelerometer and the ultrasonic sensor installed on the robots have the characteristics exposed in Table 1. In this regard, Figure 3 shows the Matlab code defining what has been commented in this paragraph.

The equations 16 – 21 were used in the simulation to make the data fusion, nevertheless the covariance matrices and those of the system were defined in the Matlab code as exposed by figure 4. In this matter, the covariance matrix of the

process noise is most of the times defined as the zero matrix because it is difficult to be defined in reality yet sometimes it is used to indicate parameters variation and or errors in the mathematical modelling process. In our case we used this matrix to define an error in the computation of the velocity as it comes from an integration of the accelerometer data which has an offset as explained before. In this sense, in this important to keep in mind that the greater the matrix Q, the slower response of the filter [Castillo-Zamora et al., 2019].

```
%% Mini Car Simulation Time and Step Time Size
tsimc=30;  % Simulation time [s]
dtc=0.001; % Time step [s]
%% Bi-Rotor Simulation Time and Step Time Size
tsimr=30;  % Simulation time [s]
dtr=0.001; % Time step [s]
%% Parameters of the Mini Car
mc=1;  % Mass [kg]
xp0=0; % Initial position [m]
xv0=0; % Initial velocity [m/s]
xd=2;  % Desired position [m]
%% Parameters of the Bi-Rotor
mr=0.5;  % Mass [kg]
zp0=2;   % Initial position [m]
zv0=0;   % Initial velocity [m/s]
zd=0.75; % Desired altitude [m]
g=9.81;  % Constant of gravity acceleration [m/s^2]
%% Mini Car Control Gains
Kpc=1; % Proportional gain
Kvc=1; % Derivative gain
%% Bi-Rotor Control Gains
Kpr=1; % Proportional gain
Kvr=1; % Derivative gain
%% Sensors Characterization
Var_a=0.0001;     % Accelerometer white noise variance
Var_abias=0.001;  % Accelerometer bias (offset) variance
bias=-0.97;       % Mean value of the accelerometer bias
Bias_dta=30*dtc;  % Bias and Uutrasonic sensor sampling time
Var_u=0.0001;     % Ultrasonic sensor white noise variance
```

Figure 3 Matlab code: Systems properties and conditions.

```
%% Mini Car Kalman Filter Parameters
Ac=[1 dtc;0 1]; % Matrix A for Kalman filter
Bc=[dtc^2/2;dtc]; % Matriz B for Kalman filter
Hc=[1 0]; % Matrix H for Kalman filter
Qc=[0 0;0 0.001]; % Covariane matrix of the process noise
Rc=Var_u; % Noise variance of the ultrasonic sensor
Pc0=[2 0;0 2]; % Initial value of covariace matrix P

%% Bi-Rotor Kalman Filter Parameters
Ar=[1 dtr;0 1]; % Matrix A for Kalman filter
Br=[dtr^2/2;dtr]; % Matriz B for Kalman filter
Hr=[1 0]; % Matrix H for Kalman filter
Qr=[0 0;0 0.001]; % Covariane matrix of the process noise
Rr=Var_u; % Noise variance of the ultrasonic sensor
Pr0=[2 0;0 2]; % Initial value of covariace matrix P

%% Running the Simulatins in Simulink
sim('SimCar',tsimc);
sim('SimBirotor',tsimr);
```

Figure 4 Matlab code: Kalman filter elements.

Figure 4 also gives evidence of the instructions used to launch the simulation in Simulink. We used to different files to complete the simulation nevertheless the structure of both files is exactly the same, the only thing that changes are the properties of the systems according to Table 1 and the equations provided in the subsections of each pedagogical platform. For such reason only the Simulink file of the mini car simulation will be explained and shown next.

Before going any further in the Simulink environment, in the “Model Parameters Configuration” option the simulation time is changed as well as the simulation type which in our case it corresponds to the fixed step time with the given sampling time in Table 1.

For different scenarios were simulated for the same system: The first one corresponded to the ideal performance of the sensors, the second one considered only the accelerometer, the third one was defined in order to use only the ultrasonic sensor and lastly, the last one merged the data coming from both sensors by means of the Kalman filter. In the four scenarios, the saturation of the motors was taken into consideration. Figure 5 depicts what is discussed in this paragraph.

In the middle of the Figure 5, surrounded by a circle, you find the clock connected to a “To Workspace block” which allows us to store the time. All the blocks of the file have a sampling time dt given in Table 1. Notice that we have used the “From” and “Goto” blocks to avoid having too many connections which would make more difficult the interpretation of the block diagram below.

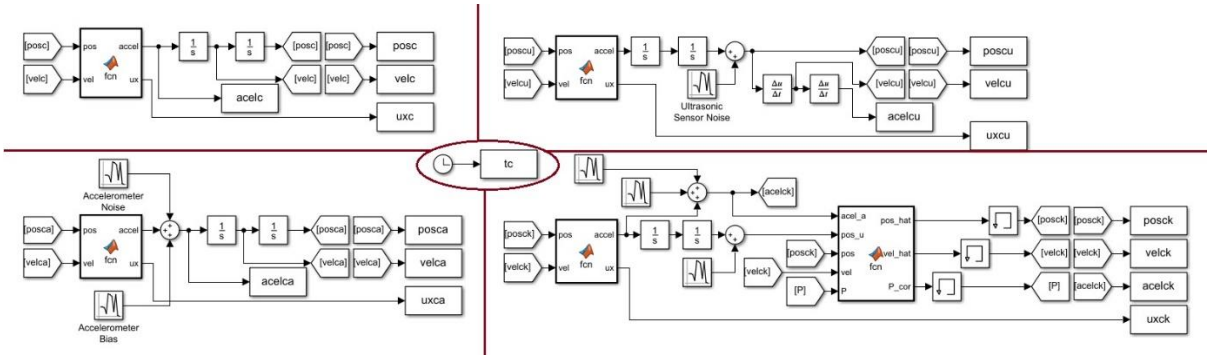


Figure 5 Simulink environment: up left corner: ideal performance of the sensors; up right corner: usage of the ultrasonic sensor only; bottom left corner: usage of the accelerometer only, and, bottom right corner: data fusion by the Kalman Filter.

The initial conditions of the system took part on the two in-chain integral blocks connected at the output of the Matlab function block. The first integrator (the one connected to the output port of the Matlab function blocks) computes the velocity and the second one does the proper for the position, correspondently the initial conditions were assigned.

The Matlab function block that appears in the four scenarios has as inputs the position and the velocity of the system. Inside this block, you can find the control equation with the saturation and the dynamics equation at the end to finally send the acceleration and the saturated control signal to the Simulink space. The code lines and the structure are exposed in Figure 6 where one may observe that, in reality, the function have several additional inputs. These four additional variables were considered as parameters of the block and not inputs, we did so by clicking over the "Edit data" option in the Matlab editor and changing the Scope option from input to parameter for xd, Kpv, Kvc and mc.

With this being said, the simulation concerning the ideal performance of the sensors can be run. For a better presentation the results are sent to Matlab where they are plotted once the simulation has finished with better quality than in Simulink.

The other 3 scenarios are simulated with base on the simulation at the up left corner block diagram in Figure 5. The block diagram located at the bottom let corner of Figure 5 differs from the ideal case by the addition of the accelerometer noise and the corresponding offset as described in Section 2. Once these two components are added to the acceleration given by the Matlab function block, it is integrated to compute the velocity and then integrated once more to obtain the position. These two computations were sent back to close the loop. The block parameters are shown in Figure 7. By defining two different sample times, we stablished that one part varies in time faster than other one.

```
function [accel,ux]= fcn(pos,vel,xd, Kpc, Kvc, mc) % Function

ux=Kpc*(xd-pos)+Kvc*(0-vel); % PD Control
ux= min(1,max(-1,ux)); % Saturation of the control signal

accel=ux/mc; % Dynamics of teh system
```

Figure 6 Mini car simulation results.

Similarly, the blocks diagram of the up right corner, which corresponds to the system when using only the ultrasonic sensor, differs from the ideal case by the addition of the ultrasonic sensor noise at the output of the integrator that computes the position. Once the position is obtained by a double integration process and the noise had been added to it, the results is derived w.r.t. time once to get the velocity and twice to compute the acceleration. The noise of this sensor is characterized as depicted in Figure 8.

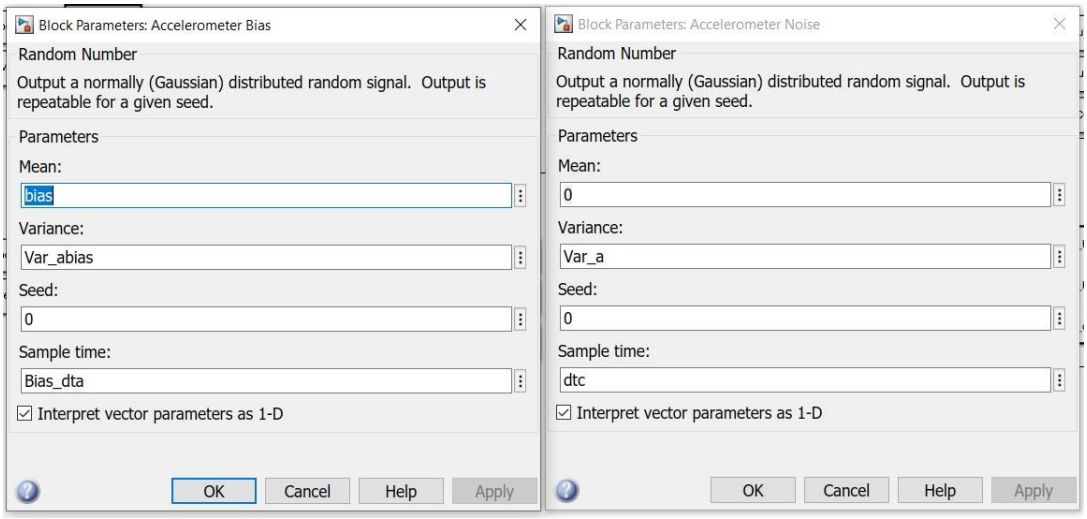


Figure 7 Accelerometer noise and offset blocks.

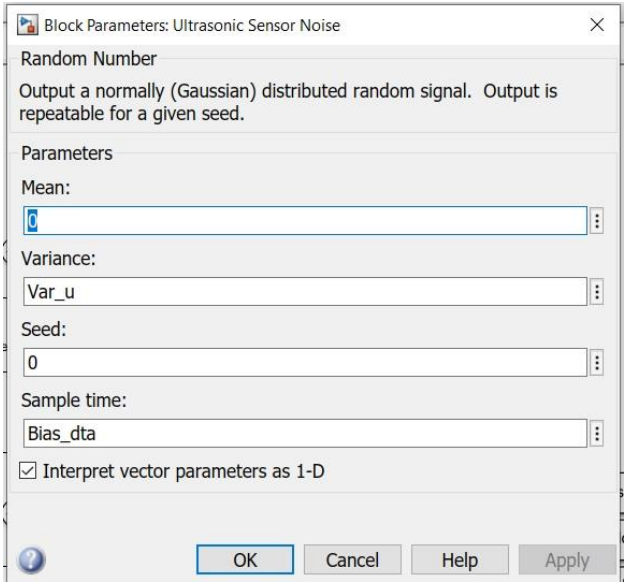


Figure 8 Ultrasonic sensor noise block.

The noisy position and velocity were used as feedback to close the system as can be appreciated in Figure 5.

To implement the Kalman filter for data fusion, we considered the accelerometer and the ultrasonic sensor as described for the previous scenarios. The signals are given as inputs to the Kalman filter Matlab function block alongside the covariance matrix as shown in Figure 5. “Memory” blocks were used to avoid the algebraic loop problem and also to initialize the filter as they contained the corresponding initial values of the position, velocity and covariance matrix. The code written inside the Kalman function block is given in Figure 9.

Once the Simulink environment was ready, the file was saved with the name “SimCar” and the simulation was run from the Matlab main file previously explained. It is worth to mention that the files should be located in the same folder.

The results of the mini car robot and the bi-rotor platform simulations are depicted in Figures 10 and 11, respectively. In both cases the positon of the vehicle, the velocity and the accelerations as well as the control inputs are depicted.

Similar conclusions can be made for both systems. One can observe, in the position plots, that when using only the accelerometer for closing the control loop, the system does not reach the desired position, i.e. it exists a steady state error which is caused (mainly) by the offset of the sensor and the saturation of the actuators in addition to the accumulative error in the integration process. To use exclusively the ultrasonic sensor seems to work in terms of position, and as expected, the system in which the Kalman filter was implemented also accomplishes the goal of reaching a given reference in a finite time.

```
1   function [pos_hat,vel_hat,P_cor] = fcn(acel_a,pos_u,pos,vel,P,Ac,Bc,Hc,Qc,Rc)
2
3   xpred=Ac*[pos;vel]+Bc*acel_a; % Equation 16
4   Ppred=Ac*P*Ac'+Qc; % Equation 17
5   K=Ppred*Hc'*inv(Hc*Ppred*Hc'+Rc); % Equation 18
6
7   y=pos_u; % Equation 19
8   xhat=xpred+K*(y-Hc*xpred); % Equation 20
9   P_cor=(eye(2)-K*Hc)*Ppred; % Equation 21
10
11  pos_hat=xhat(1); % Selecting the estimated position
12  vel_hat=xhat(2); % Selecting the estimated velocity
```

Figure 9 Mini car simulation results.

The velocity plots show the disadvantage of using only the ultrasonic sensor as the computed velocity becomes noise and its magnitude is big in comparison with the velocity in the ideal scenario. The Kalman filter provides a noisy velocity that tracks the desired behavior. The velocity obtained only by the integration of the accelerometer signal behaves far from the desired response. In this sense, the Kalman Filter provides a better velocity estimation.

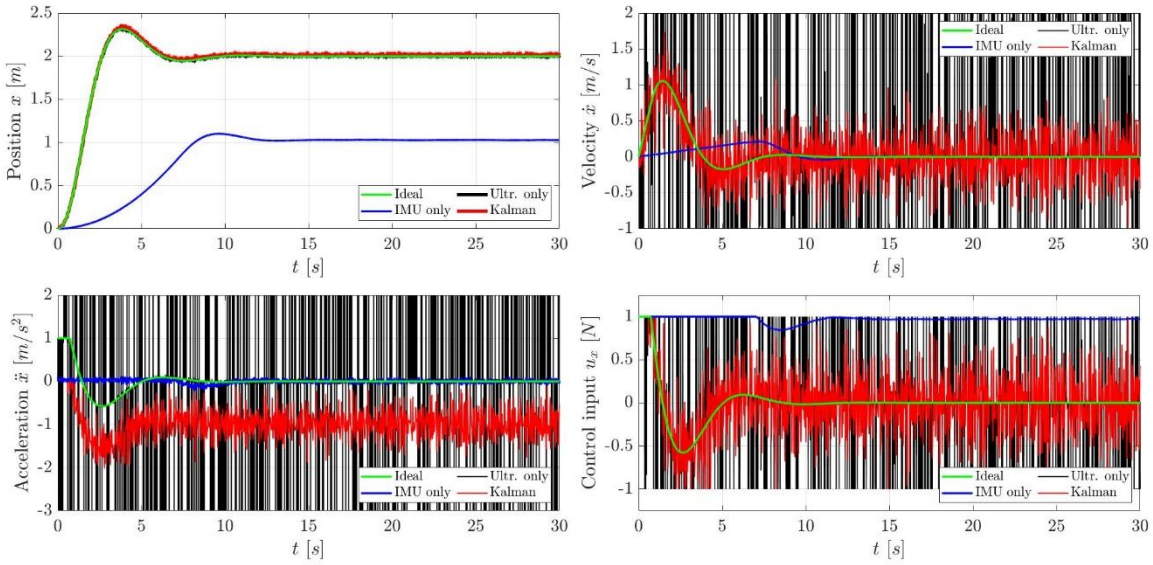


Figure 10 Mini car simulation results.

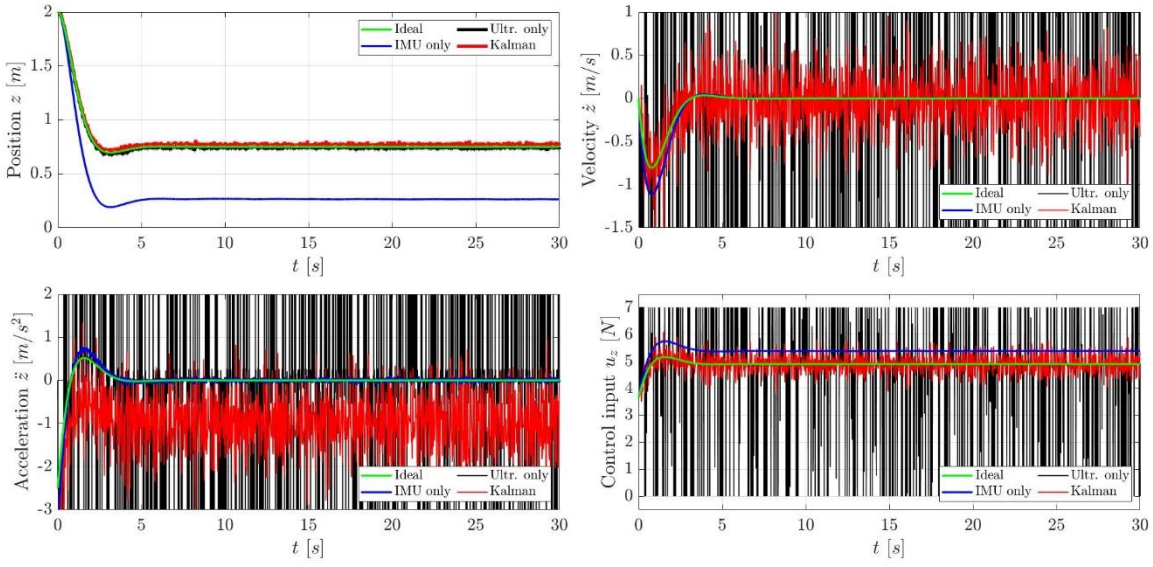


Figure 11 Bi-rotor simulation results.

The acceleration plots show a magnification of the acceleration in the case where only the ultrasonic sensor is used. On the other hand, the acceleration given by the accelerometer is clearly affected by the offset and the noise. For the case of the Kalman Filter implementation, we can conclude that even when the acceleration provided by the accelerometer is not the real one, the data fusion technique does a correct estimation of the position and the velocity, making the system reach the desired position even with wrong acceleration measurements.

The influence saturation of the actuators is more evident in the case of the mini car robot as depicted by the corresponding plots, however the system is able to overcome such issue when implementing the data fusion by means of the Kalman Filter.

4. Conclusions

A pedagogical approach to Kalman Filtering was given and explained considering two examples where such technique was use to illustrate the data fusion algorithm. In addition, the characterization of an ultrasonic sensor and an accelerometer was established and can be considered in future simulations for students. The mathematical models of the vehicles were developed and additional systems suggestions were proposed. It is expected the usage of the in-here studied technique to improve the results of academic projects in engineering formation.

6. Bibliografía y Referencias

[1] Sasiadek, J. Z., & Hartana, P. (2000). Sensor data fusion using Kalman filter. In Proceedings of the Third International Conference on Information Fusion (Vol. 2, pp. WED5-19). IEEE.

[2] Carullo, A., & Parvis, M. (2001). An ultrasonic sensor for distance measurement in automotive applications. IEEE Sensors journal, 1(2), 143.

[3] Gan, Q., & Harris, C. J. (2001). Comparison of two measurement fusion methods for Kalman-filter-based multisensor data fusion. IEEE Transactions on Aerospace and Electronic systems, 37(1), 273-279.

[4] Gao, J. B., & Harris, C. J. (2002). Some remarks on Kalman filters for the multisensor fusion. Information Fusion, 3(3), 191-201.

[5] Castillo-Zamora, J. J., Camarillo-Gómez, K. A., & Pérez-Soto, G. I. (2015). Diseño de un Control PID de Posición para un Cuadricoptero V-tail Basado en la Teoría de Control de Robots Manipuladores. XVII Congreso Mexicano de Robótica (COMRoB).

[6] Awasthi, S., & Joshi, A. (2015). MEMS accelerometer based system for motion analysis. In 2015 2nd International Conference on Electronics and Communication Systems (ICECS) (pp. 762-767). IEEE.

[7] Castillo-Zamora, J. J. (2016). Analysis and Comparison of Dynamics and Control Simulations of a V-tail Quadcopter and a Mini AUV. Master Thesis, Tecnológico Nacional de México en Celaya.

[8] Escareño, J., Castillo, J., Abassi, W., Flores, G., & Camarillo, K. (2017, October). Navigation strategy in-flight retrieving and transportation operations for a rotorcraft mav. In 2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS) (pp. 7-12). IEEE.

[9] Liggins II, M., Hall, D. L., & Llinas, J. (2017). Multisensor data fusion. In Handbook of multisensor data fusion. CRC press.

[10] Castillo-Zamora, J. J., Camarillo-Gómez, K. A., Pérez-Soto, G. I., & Rodríguez-Reséndiz, J. (2018). Comparison of PD, PID and sliding-mode position controllers for V–tail quadcopter stability. IEEE Access, 6, 38086-38096.

[11] Zheng, Z., Qiu, H., Wang, Z., Luo, S., & Lei, Y. (2019). Data fusion based multi-rate Kalman filtering with unknown input for on-line estimation of dynamic displacements. Measurement, 131, 211-218.

[12] Castillo-Zamora, J. J., Escareño, J., Alvarez, J., Stephant, J., & Boussaada, I. (2019). Disturbances and Coupling Compensation for Trajectory Tracking of a Multi-link Aerial Robot. In 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT) (pp. 738-743). IEEE.