IEC
Institute of Electronics and Computer

Journal of Artificial Intelligence and Systems, 2025, 7, 89-111
https://iecscience.org/journals/AIS
ISSN Online: 2642-2859

Neural Network-Based Estimation of Robot Trajectory Using Kalman Filter Data Fusion from Encoders and Stereo Cameras

Luciana Claudia Martins Ferreira Diogenes

Frutal - MG, Brazil
*Corresponding Author: Luciana Claudia Martins Ferreira Diogenes, Email: lucianafem@yahoo.com.br

How to cite this paper: Luciana Claudia Martins Ferreira Diogenes (2025). Neural Network-Based Estimation of Robot Trajectory Using Kalman Filter Data Fusion from Encoders and Stereo Cameras. Journal of Artificial Intelligence and Systems, 7, 89–111. https://doi.org/10.33969/AIS.2025070106.

Received: October 31, 2024
Accepted: November 24, 2025
Published: December 2, 2025

Abstract

This paper explores autonomous robot navigation using two types of sensors: encoders and a pair of cameras. The robot is tasked with navigating through obstacles of varying color intensities to reach a designated goal: a door. The robot's trajectory is determined using a Python-based algorithm that employs sensor fusion through the Kalman Filter to estimate the robot's position accurately. To investigate the feasibility of training a neural network with trajectory data obtained from the Kalman Filter, a separate Python script is developed to assess the accuracy between the estimated positions and those predicted by the neural network. The training results demonstrate that the neural network can be effectively trained, achieving a relatively high level of precision in position estimation. This research underscores the potential of combining sensor fusion and machine learning techniques to enhance the navigation capabilities of autonomous robots in dynamic environments.

Keywords

Neural networks, Kalman filter, navigation systems, autonomous robots, machine learning

1. Introduction

The Kalman filter has a wide range of technological applications. One of its most prevalent uses is in the guidance, navigation, and control of vehicles, particularly in aviation and space exploration. Additionally, the Kalman filter is a well-established concept in time series analysis, relevant in areas like signal processing and econometrics. It is also a key topic in robotic motion planning and control, often integrated into trajectory optimization processes. This algorithm operates in a two-step cycle. In the prediction phase, the Kalman filter generates estimates of the current state variables along with their associated uncertainties. Once the next measurement—inevitably subject to some error and random noise —is obtained, these estimates are refined through a weighted average, where greater weight is assigned to estimates with higher confidence. The algorithm is recursive and can function in real time, utilizing current input measurements and previously computed state and uncertainty matrices without needing any historical data.

For example, consider the challenge of accurately determining a truck's location [1]. The truck may be equipped with a GPS unit that provides a position estimate accurate to within a few meters. However, this GPS data is likely to be noisy, with readings fluctuating while remaining close to the true position. Furthermore, since the truck is expected to adhere to the laws of physics, its position can also be estimated by integrating its velocity over time, which is tracked by monitoring wheel revolutions and steering wheel angles. This method is known as dead reckoning. While dead reckoning typically yields a smooth position estimate, it tends to drift over time due to cumulative small errors.

In this scenario, the Kalman filter operates in two phases: prediction and update. During the prediction phase, the truck’s previous position is adjusted according to the laws of motion (the dynamic or "state transition" model). This phase not only calculates a new position estimate but also updates the covariance. The covariance may be related to the truck's speed, reflecting increased uncertainty in dead reckoning estimates at higher speeds, while remaining more accurate at slower speeds. In the update phase, a position measurement from the GPS is taken. This measurement comes with its own uncertainty, and its covariance in relation to the previous position affects how much the new measurement influences the updated prediction. Ideally, as the dead reckoning estimates drift away from the actual position, the GPS measurement should help correct the estimate without causing excessive fluctuations or noise.

Faced with a problem like this of controlling navigation in a truck, it can use this same method for the application of robots.

An example of the application of autonomous robots in industry is robots in logistics, which include, for example, autonomous machines used to automate goods flows, maximize safety, and increase the productivity of warehouse operations. The entry of new technologies into any of the warehouse operations, such as the receipt of goods, storage, inventory management, order preparation and the dispatch of goods, has made these robots gain greater prominence. These solutions operate with complete autonomy to perform functions such as transferring goods between two points, preparing orders or storing products on racks.

In [2] the entire system for an autonomous robot navigation in 2D was described containing some objects was developed with MATLAB. Based on this work and using the Python programming language, this work could be developed with the additional of a second program that uses neural networks to estimate the trajectory of a part of the robot's navigation.

Consider that for this work a method similar to truck navigation control was applied [1], but instead of a GPS unit, a pair of cameras are placed on the base of the robot to obtain location information, and at the same time, the laws of motion of physics are used to estimate its position through calculations of the movement of the wheels. In addition, the doctoral thesis that was developed by [2] was taken as the basis of this work also because applied Kalman's filter to the robot's navigation. In this work there is an algorithm developed in Python language containing a neural network architecture is used to predict a part of the trajectory. In this way, it is possible to compare the robot's position values estimated by the Kalman Filter with the values predicted by the neural network.

2. Related Work

In [3], the study was conducted in the context of humanoid robotics and addresses the challenge of tracking the position and velocity of objects in translational sliding motion, using only tactile observations. The research proposed a differentiable Extended Kalman Filter (dEKF) trained to track the state of an object during sliding, utilizing data from tactile sensors. Experiments performed with various objects on the iCub humanoid robot platform demonstrated that the proposed method achieved an average position tracking error of approximately 0.6 cm. Additionally, the estimated state of the object could be used to make control decisions based solely on tactile feedback. The paper began with an introduction to the problem of object perception in autonomous manipulation, reviewing related work that employs deep neural networks to estimate object pose from RGB-D images. It then discusses the specific challenges of tracking objects subjected to sliding and presents a literature review on slip detection and prediction using tactile sensors. The proposed method models object sliding as a differentiable Kalman filtering problem, learning motion and measurement models from real data obtained through visual feedback. The filter was implemented as a neural network that learns to relate tactile measurements to the state of the moving object. Experimental results showed that the proposed filter achieves accurate performance in tracking the position and velocity of the object during sliding. Moreover, the paper included a section on related work, details about the materials and methods used, as well as a description of the training procedure and software implementation. The differentiable Extended Kalman Filter (dEKF) is a version of the extended Kalman filter that can be trained using machine learning methods. The extended Kalman filter is a state estimation tool that handles nonlinear systems, allowing for the estimation of state variables from noisy and incomplete measurements. The differentiable version enables movement and measurement models to be learned from experimental data rather than being derived from first-order physical principles. In the context of this document, the dEKF was employed to track the position and velocity of objects undergoing translational sliding, using only tactile observations. This is particularly useful in humanoid robotics, where the ability to perceive and control object slipping is essential for autonomous manipulation. The dEKF is trained with tactile sensor data and can provide an accurate estimate of the object's state, allowing the robot to make control decisions based solely on tactile feedback.

It was reported in [4] that Kalman filter (KF) is a widely used estimation algorithm for many applications. However, it faces challenges due to imperfect mathematical models, dynamic environments, and inaccurate parameters. Artificial intelligence (AI) techniques have been applied to improve KF's performance. The reviewed 55 papers proposed KF with AI techniques to improve performance, categorized into four groups: tuning parameters, compensating errors, updating state vector or measurements, and estimating pseudo-measurements. The paper discusses the challenges, research gaps, and future research directions, emphasizing the need for systematic validation, overcoming nonlinearity, and combining AI-based approaches for performance improvement. Kalman Filter (KF) is a widely used recursive estimator, with variations such as extended Kalman filter (EKF) and unscented Kalman filter (UKF). The measure model for KF can be denoted as a state-space model, and it can be divided into two main steps: prediction and update. However, KF can be degraded in many real applications due to nonlinearity. To overcome this, EKF and UKF are widely used. Neural Network (NN) is another popular technique, with feedforward neural network (FFNN) being commonly used. FFNN is composed of input, hidden, and output layers, and outputs are obtained by weights connecting the layers. The paper also discusses the application of AI techniques such as neural networks, recurrent neural networks (RNN), and reinforcement learning in various fields, emphasizing the potential of combining KF with AI techniques for performance improvement. The text discusses various research papers on the integration of artificial intelligence techniques with Kalman filters for different applications such as vehicle roll angle estimation, water level forecasting, speech enhancement, and state estimation. It also mentions a review paper on the topic.

In was proposed [5] an optimized Kalman filter approach based on an improved Gray Wolf algorithm (IGWO-OKF) to improve the prediction accuracy of target tracking. It optimizes the process noise covariance matrix and observation noise covariance matrix in Kalman filter. The approach is applied to target tracking and shows low error, high accuracy, and good prediction effect. The paper also discusses the application of Kalman filtering algorithm in landslide monitoring and the combination of Gray Wolf algorithm with Kalman filtering to improve state estimation and optimization tasks in complex problems and real-time systems. The paper presented the improved Gray Wolf algorithm and its advantages, such as simplicity, global search capability, fast convergence rate, and parameter adaptivity. It also discusses the application of the improved Gray Wolf algorithm in optimizing Kalman filtering. The paper compared the performance of the proposed IGWO-OKF model with the traditional variance-compensated adaptive Kalman filtering algorithm in deformation monitoring of an open-pit mine. The experimental results showed that the IGWO-OKF model achieves faster convergence to actual observations and less volatility, indicating stronger convergence ability, better stability, and improved filtering results and deformation prediction accuracy compared to the traditional algorithm. The paper concluded by suggesting the application of the proposed filtering model in landslide monitoring and the need for further improvement in the measurement update and optimization of the Kalman filter.

It was presented in [6] a proposal and tests a speed observer and controller for DC motors equipped with low-resolution encoders, specifically for mini robots with differential drive. The system was implemented in the microcontroller (ESP32) and demonstrated the ability to control the speed of motor-wheel pairs of differential drive robots (with low-precision rotary encoders) to reduce their asymmetries. Experimental results showed that the system could control the speed of the motor-wheel pairs of differential drive robots (with low-precision rotary encoders) to reduce their asymmetries. A control scheme with two control strategies, Feedforward/Backward, was used, with the Kalman filter as the speed estimator to reduce the sensors' quantization error. The Kalman filter was used as the speed estimator to reduce the sensor's quantization error.

The study addressed in [7] the challenges in mobile robot navigation, proposing the use of LIDAR sensors and extended Kalman filter to correct cumulative errors in inertial navigation systems. It was emphasized the importance of precise localization and demonstrated the successful application of the EKF technique to correct wheel odometer's cumulative error with the assistance of the LIDAR sensor.

In [8] presented the design and tuning of an extended Kalman filter for identifying robotic systems. What is most used in robotic identification models is the use of the inverse dynamic model and the least squares method, which often require fine-tuned filtering or estimation of position, velocity, acceleration, and torque to avoid distortions. The cutoff frequency of the low-pass filter used must be well chosen, which is not always a trivial task. The authors proposed the use of an extended Kalman filter to reduce noise at the measured position and estimate velocity and acceleration. These estimates could then be sent to the controller to further reduce noise in the control torque. The effect of adjusting this filter was examined and the presented approach is validated through simulations and experiments in a one degree of freedom system.

It was proposed in [9] a soft sensor implementation using an unscented Kalman filter to validate experimentally challenging variables like fluid viscosity, productivity index, and production flow rate in a pilot plant equipped with an Electrical Submersible Pump (ESP) and a comprehensive supervisory system for data collection and registration. ESPs operate in harsh conditions, where uncertainties and failures in variables are common in both onshore and offshore oil fields. Through experimental tuning, the soft sensor effectively brought these variables within a 95% confidence interval during continuous ESP operation over a 26-hour experiment.

In [10], the Kalman filter was a minimum-variance estimation method for dynamic systems, particularly in target tracking. It had gained attention due to its real-time, fast, efficient, and strong anti-interference properties. The Kalman filter has been widely applied in fields such as orbit calculation, target tracking, navigation, integrated navigation, dynamic positioning, sensor data fusion, microeconomics, digital image processing, pattern recognition, image segmentation, and image edge detection. The Kalman filter mainly includes Kalman filter (KF), Extended Kalman filter (EKF), and Unscented Kalman filter (UKF). Kalman filter is a linear optimal status estimation method, known as one of the most famous Bayesian filter theories. It was defined as a linear representation of status and observation equations, with the prior and posterior estimations derived from the status transition equation at the moment of k-1. The Kalman filter equations was calculated by calculating the linear combination of a priori estimation and weighted difference between true measurements and measured forecast.

Given its extensive use in robotics, the Kalman filter has garnered significant attention from researchers [11]. This study reviewed recent modifications made to the algorithm over the past years, addressing issues such as consistency, convergence, and accuracy. Six decades since its inception, the Kalman filter remains pivotal in autonomous navigation, robot control, trajectory tracking, and various other applications. Furthermore, this review analyzed and compared the specific characteristics of each modification made to the filter.

This study shown in [12] introduced a method for estimating the position of a self-driving solar panel-cleaning mobile robot using line counts, wheel encoder information, and inertial sensor data. Two adjusted threshold values and offsets were introduced based on the robot's speed to achieve accurate line counts. Inertial measurement unit (IMU) signals were used to determine if a line is horizontal or vertical, depending on the robot's movement direction on the panel. When the robot is positioned between lines on the panel, more precise location estimation was necessary. To tackle this challenge, the extended Kalman filter was integrated with IMU data and encoder information, significantly enhancing position estimation. This integration achieved an RMSE accuracy value of up to 0.089 m, notably at a relatively high speed of 100 mm/s. This margin of error was almost half that of the vision-based line-counting method.

The study conducted in [13] focused on developing a navigation system for a terrestrial mobile robot to autonomously follow a predefined path. This work reviewed the State of the Art in Kalman filters and navigation systems, describing developed algorithms, implementation methods, and presenting relevant results from the system's application. The control utilized the robot's sensors (IMU, GPS, and odometry), processing information using Matlab and C# on a PC, with Wireless networks for communication. Odometric sensor data was combined to estimate the robot's instantaneous position, and error relative to the position obtained from GPS was determined. These errors were filtered using linear Kalman filters and reintegrated into the position estimate. The goal was to achieve an optimal discrete estimate of the robot's instantaneous position, crucial for precise autonomous navigation. Evaluation tests demonstrating the robot's ability to automatically traverse a predefined path were presented, confirming the validity of the work and the functionality of the algorithms.

For [14] internal tank inspection could be dangerous and expensive. Teleoperated robotic systems offer a safer and more efficient alternative, often without the need to empty the tank. To locate defects, the robot must know its relative position inside the tank. Self-localization is essential, but robots with magnetic wheels face challenges like wheel slippage and magnetic interference. A Kalman filter was developed to locate a robot with four fixed magnetic wheels, using inertial sensors and odometry, discarding magnetic measurements due to interference. Simulations and tests confirmed the filter's effectiveness.

3. Theory and Calculation

To help understand how robot position calculations are calculated using a pair of cameras, consider Figure 1. The robot in the picture is displaced from its initial position (0.0) m and with a theta direction. In front of it is located a pair of cameras a and b. A point located in the plane P (x_i, y_i) is projected onto the two cameras with the x coordinates being r_i and s_i, respectively. The focus of the cameras is given by f. The cameras are separated by a distance t. One vector $\mathbf{x}_a$ is formed between the robot and camera a and another vector $\mathbf{x}_b$ is formed between the robot and camera b.

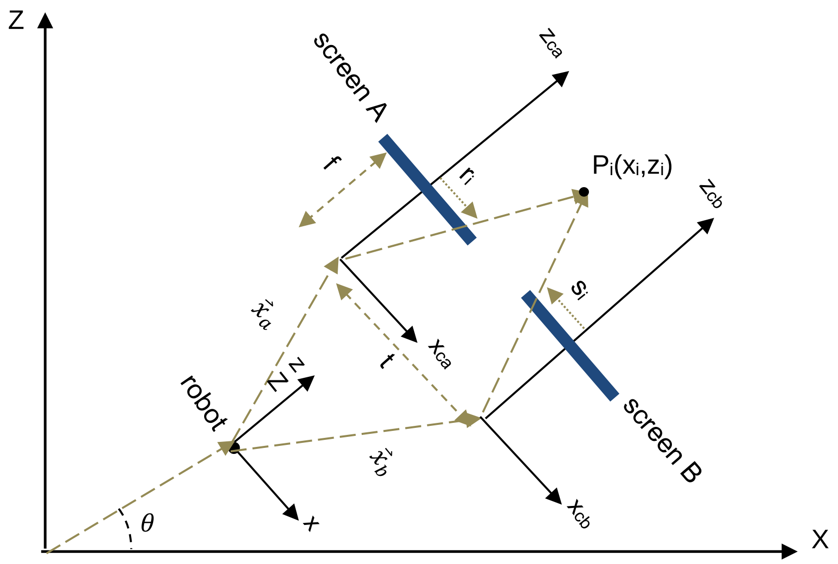

Figure 1. Position of the robot in the world and the position of the cameras in relation to the robot and the world.

The location of the point P_i (x_i, z_i) in the robot's reference frame can be recalculated using the positions of that same point in the two images. That way, consider the Equation 1 :

$$\boldsymbol{x}_i = \begin{bmatrix} x_i \\ z_i \end{bmatrix} = \frac{1}{(r_i - s_i)} \begin{bmatrix} r_i \\ f \end{bmatrix} \tag{1}$$

The uncertainties of the point P_i are given by Equation 2:

$$\mathbf{dx}_i = \begin{bmatrix} dx_i \\ dz_i \end{bmatrix} \tag{2}$$

And they can be rewritten as follows the Equation 3:

$$\mathbf{dx}_i = \boldsymbol{B}_i \boldsymbol{dr}_i \tag{3}$$

Where the matrix B_i is given by Equation 4:

$$\boldsymbol{B}_i = \frac{\boldsymbol{z}_i^2}{\boldsymbol{f}\boldsymbol{t}}\begin{bmatrix} -\frac{\boldsymbol{s}_i}{\boldsymbol{f}} & \frac{\boldsymbol{r}_i}{\boldsymbol{f}} \\ \boldsymbol{-1} & \boldsymbol{1} \end{bmatrix} \tag{4}$$

and the vector **dr**ᵢ is given by Equation 5:

$$\mathbf{dr}_i = \begin{bmatrix} dr_i \\ ds_i \end{bmatrix} \tag{5}$$

The robot must choose a passage portal between two objects and in this way, consider two points in the plane, P_i and P_j as is shown in Figure 2, being:
β: angle formed between the line $\overline{P_i P_j}$ and x axis.
γ: angle formed between the line $\overline{P_i P_j}$ and X axis.
α: angle formed between the line $\overline{P_i P_j}$ and z axis.
Z_θ: angle formed between z axis and X axis.

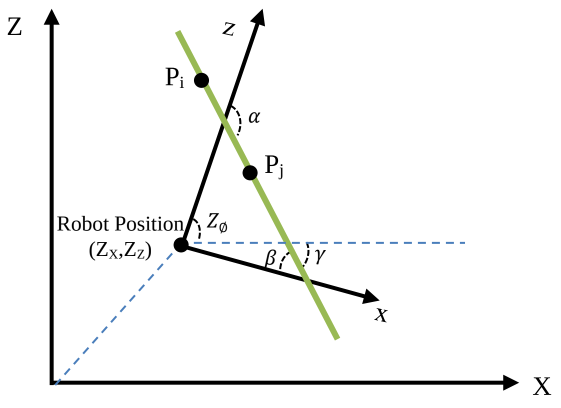

Figure 2. Image used to calculate the position of the robot through points P_i and P_j.

Through the Figure 2, the robot's Z position in the world can be calculated by simple vector sum calculations, obtaining the Equation 6:

$$\boldsymbol{Z} = \begin{bmatrix} Z_x \\ Z_z \\ Z_\emptyset \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\left[\boldsymbol{P}_i + \boldsymbol{P}_j - \boldsymbol{R}^T(\boldsymbol{x}_i + \boldsymbol{x}_j)\right] \\ \alpha - \gamma \end{bmatrix} \tag{6}$$

The Equation 7 shows rotation matrix R:

$$\boldsymbol{R} = \begin{bmatrix} sin(Z_\emptyset) & -cos(Z_\emptyset) \\ cos(Z_\emptyset) & sin(Z_\emptyset) \end{bmatrix} \tag{7}$$

Observing the Figure 2, the angle α and β are calculated as follows:

$$\alpha = \frac{\pi}{2} + \beta \tag{8}$$

$$\beta = arctan\left(\frac{z_i - z_j}{x_j - x_i}\right) \tag{9}$$

With x_i, z_i, x_j and z_j being the x and z coordinates of points P_i and P_j, as seen from the robot’s reference frame.

By deriving the robot’s state vector Z, the Equation 10 shows how is possible to obtain the uncertainty dZ:

$$\boldsymbol{dZ} = \begin{bmatrix} dZ_x \\ dZ_z \\ dZ_\emptyset \end{bmatrix} = \begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{C} \end{bmatrix} \boldsymbol{dr}_{ij} \tag{10}$$

Where:

$$\boldsymbol{D} = -\frac{1}{2}\left[\frac{\partial \boldsymbol{R}^T}{\partial Z_\emptyset}(\boldsymbol{x}_i + \boldsymbol{x}_j)\boldsymbol{C} + \boldsymbol{R}^T[\boldsymbol{B}_i \quad \boldsymbol{B}_j]\right] \tag{11}$$

$$\boldsymbol{C} = \frac{cos^2(\beta)}{x_j - x_i}[tan(\beta) \quad 1 \quad -tan(\beta) \quad -1] \tag{12}$$

$$\mathbf{dr}_{ij} = \begin{bmatrix} dr_i \\ dr_j \end{bmatrix} \tag{13}$$

Some of the equations used in the Kalman filter use the covariance of the state noise. Thus, considering that the uncertainty of the robot's position is given by a factor dZ and that the uncertainties of the cameras are also Gaussian noise, the covariance matrix of dZ is given by Equation 14:

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{C} \end{bmatrix} \boldsymbol{E}[\boldsymbol{D}^T \quad \boldsymbol{C}^T] \tag{14}$$

Where:

$$\boldsymbol{E} = \nu^2 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

v : is the standard deviation of computer vision uncertainty.

Consider that the model of this work, where a robot navigated through an environment, the ideal state of the robot $\widetilde{\boldsymbol{X}}_k$ is given by Equation 16 ou 17:

$$\widetilde{\boldsymbol{X}}_k = \begin{bmatrix} \tilde{X}_{xk} \\ \tilde{X}_{zk} \\ \tilde{X}_{\emptyset k} \end{bmatrix} \tag{16}$$

$$\widetilde{\boldsymbol{X}}_k = \widetilde{\boldsymbol{X}}_{k-1} + \widetilde{\boldsymbol{G}}_{k-1}\widetilde{\boldsymbol{u}}_k + \widetilde{\boldsymbol{q}}_k \tag{17}$$

where:
$\tilde{X}_{\emptyset k}$: angle that the robot's direction makes from the x axis.
$\widetilde{\boldsymbol{X}}_k$: real state of the robot at instant k,
$\widetilde{\boldsymbol{q}}_k$: the process noise, assumed to be sampled from a Gaussian distribution,
$\widetilde{\boldsymbol{G}}_{k-1}$: matrix is given by Equation 18:

$$\widetilde{\boldsymbol{G}}_{k-1} = \begin{bmatrix} \frac{\pi r_d}{N_{res}} cos(\tilde{X}_{\emptyset k-1}) & \frac{\pi r_e}{N_{res}} cos(\tilde{X}_{\emptyset k-1}) \\ \frac{\pi r_d}{N_{res}} sin(\tilde{X}_{\emptyset k-1}) & \frac{\pi r_e}{N_{res}} sin(\tilde{X}_{\emptyset k-1}) \\ \frac{2\pi r_d}{N_{res}} & -\frac{2\pi r_e}{N_{res}} \end{bmatrix} \tag{18}$$

$\widetilde{\boldsymbol{u}}_k$: input rotation or odometry measurement given by Equation 19 :

$$\widetilde{\boldsymbol{u}}_k = \begin{bmatrix} N_{dk} \\ N_{ek} \end{bmatrix} \tag{19}$$

Assuming that the state uncertainty $\mathbf{d}\widetilde{\boldsymbol{X}}_k$ has within it the encoder measurement uncertainties that are assumed to be Gaussian noise, the covariance matrix of the uncertainty **Q** is given by Equation 20:

$$\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{d}\boldsymbol{L}^T \tag{20}$$

where:

$$\mathbf{d}\boldsymbol{X}_k = \boldsymbol{L}_k\widetilde{\boldsymbol{u}}_k \tag{21}$$

$$\boldsymbol{L}_k = \frac{\partial \widetilde{\boldsymbol{G}}_k}{\partial \tilde{X}_{\emptyset k}} \widetilde{\boldsymbol{u}}_k \begin{bmatrix} \frac{2\pi r_d}{N_{res}} & -\frac{2\pi r_e}{N_{res}} \end{bmatrix} + \widetilde{\boldsymbol{G}}_k \tag{22}$$

$$\boldsymbol{d} = \omega^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{23}$$

ω : is the standard deviation of odometry uncertainty.

Sensor fusion is essential for the navigation of autonomous robots and with the fusion of encoders and a pair of cameras is possible to have a robust and realistic navigation model that it could later be used on a real robot. However, the state measurements of this dynamic model from each sensor contain inaccuracies and a solution to improve this measurement would be to use the Kalman filter. This filter was developed by Rudolf Kalman in 1960 and is based on dynamic discretized linear systems in the time domain as showed in [15, p.7-8], [16, p.290-292], [17, p.12] and [18, p.15 – 17]. The model for the Kalman filter assumes that the real state at time k is obtained through the state at time (k – 1) according to the Equation 24:

$$\boldsymbol{X}_k = \boldsymbol{F_k}\boldsymbol{X}_{k-1} + \boldsymbol{G}_{k-1}\boldsymbol{u}_k + \boldsymbol{q}_k \tag{24}$$

where:

X$_k$: is the real pose of the robot at instant k,

F$_k$: k is the transition matrix taking the state **X**$_k$ from time k-1 to time k,

X$_{k-1}$: is the real pose of the robot at instant k-1,

u$_k$: the input rotation or odometry measurement,

G$_{k-1}$: control matrix,

q$_k$: the process noise, assumed to be sampled from a Gaussian distribution with zero mean and covariance **Q**$_k$ as is showed in Equation 25.

$$p(\boldsymbol{q}_k) \sim N(0, \boldsymbol{Q}_k) \tag{25}$$

The Kalman Filter considers that the measure **z**$_K$ is linearly related to the state of the system **X**$_k$, mathematically saying as showed in Equation 26:

$$\boldsymbol{Z}_k = \boldsymbol{H}\boldsymbol{X}_k + \boldsymbol{s}_k \tag{26}$$

where:

H: observation matrix,

s$_k$: measurement noise.

Since the measurement provided by computer vision is not perfect and it is obtained from Gaussian distribution with zero mean and covariance **S**$_k$, as appears in Equation 27:

$$p(\boldsymbol{s}_k) \sim N(0, \boldsymbol{S}_k) \tag{27}$$

The Discrete Kalman Filter is modeled on two operations basics: predict and update. In the prediction phase, make a prediction of a state, based on some previous values and model. Subsequently, obtain the measurement of that state, from sensor. In the end, it is updating a prediction, based on errors.
In the prediction phase, the Equations 28 and 29 are used:

$$\boldsymbol{X}_k^- = A\widehat{\boldsymbol{X}}_{k-1} + \boldsymbol{G}_{k-1}\boldsymbol{u}_k \tag{28}$$

$$\boldsymbol{P}_k^- = \boldsymbol{A}\widehat{\boldsymbol{P}}_{k-1}\boldsymbol{A} + \boldsymbol{Q}_k \tag{29}$$

where:
$\boldsymbol{X}_k^-$: a-priori state estimate of the system,
$\widehat{\boldsymbol{X}}_{k-1}$: previous optimal state estimate,
$\boldsymbol{P}_k^-$: estimated a-priori error covariance matrix,
$\widehat{\boldsymbol{P}}_{k-1}$: a-posteriori estimate error covariance.

In the update phase, it becomes necessary to compute the optimal Kalman gain matrix $\mathbf{K}_k$. Consider the Equation 30:

$$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_k^-\boldsymbol{H}^T + \boldsymbol{S}_k)^{-1} \tag{30}$$

The a-posteriori (or optimum) estimate of the system's state $\widehat{\boldsymbol{X}}_k$ in Equation 31, becomes:

$$\widehat{\boldsymbol{X}}_k = \boldsymbol{X}_k^- + \boldsymbol{K}_k(\boldsymbol{Z}_k - \boldsymbol{H}\boldsymbol{X}_k^-) \tag{31}$$

The updated error covariance matrix $\widehat{\boldsymbol{P}}_k$ is calculated by Equation (32):

$$\widehat{\boldsymbol{P}}_k = (\mathbf{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^- \tag{32}$$

It can be observed that Equation 17 is similar to 24 and thus, the Kalman filter can be applied.

4. Procedure

The work was divided into two parts that correspond to two different scripts; both being developed in the Python language. The first script described the robot's navigation by applying the fusion of data from the two sensors: encoders and camera pairs, in the Kalman filter. As the robot took a step, the robot's location can be obtained: by considering an optimal trajectory (without uncertainties), the real (considering the encoders) and the estimated (via Kalman's filter).

The second script took the data obtained from the first, only from the estimated location. A part of this location was used as a training and the other part for testing.

Training values were used to train a neural network whose architecture is built with a few intermediate layers. The test data was used to compare with the output values provided through the neural network. In this way, these values were compared to find out whether the neural network could be used to estimate a part of the robot's trajectory.

4.1. How to Estimate Position Using Kalman Filter

The robot aimed to navigate between objects within a rectangular environment with dimensions 8 x 10 m and reach a door (the goal) located at point (7.9, 9.8) m. In this environment, twelve circular objects with a radius of 0.2 m are located, each of which has a different color intensity. The robot was initially located at point (0.0) m with its direction pointed towards the goal (0.89 rad), where it made a 180° scan in the field of view, to find a passage portal, before reaching the objective. The Table 1 shows the initial input parameters of the program developed in Python. The goal may not be seen directly by the robot, so it becomes necessary to find a stopping point for the robot to try to find you again. Up to this point, the robot tries to make a straight path where at each step a position measurement is performed via odometry and computer vision, these two measurements are inserted into the Kalman filter to know the estimated position. Upon reaching the portal, it scans 180° again and, in this way, the process it is repeated until the goal is reached.

These portals were found through a distance between two objects, called the margin of safety, located by a pair of cameras. In the robot were located two cameras, a and b, where both process the images of objects in the two-dimensional environment into one-dimensional images, equivalent to a step function. The position of the robot based on computer vision aimed to find out what the same steps are in both cameras, what corresponds to the same object, and the position of the extremes of that step on the x-axis in each camera. There was a need to find at least two points corresponding to the extremes adjacent to two different steps, in both cameras, and points closer to the cameras mean less location error. These two dots signify two different objects, it was defined a point between them, and in this way the portal was defined.

From this simulation, three types of graphs can be obtained that show the robot's navigation: the ideal, the actual and the estimated navigation.

In ideal navigation, the movement is perfectly straight, without encoder inaccuracy, for example. In actual navigation, the inaccuracies of the encoders are added and the robot oscillates its trajectory with the ideal navigation as a reference. In the estimated navigation, there is navigation that unites the information from the two types of sensors: encoders and pair of cameras, to seek a more accurate navigation than the real one. The inaccuracy of the position estimated by the encoders is incremental, and the inaccuracy of the position estimated by the cameras is lower when the objects are closer.

Table 1. Parameters used in the Python program to estimate the robot's position via Kalman Filter.

Starting Position Vector (m, m, rad)	[0, 0, 0.8923]
Goal (m)	[7.9, 9.8]
Wheel radius (m)	0.1
Distance between wheels (m)	0.2
Number of Encoder Steps per Revolution (Nres)	100
Number of Encoder Steps Considered in Motion (N)	30
Standard Deviation for Odometry Error (σ) (m)	1.3
Desvio Padrão para o Erro de Câmera ($\bar{\sigma}$) (m)	$5.7*10^{-7}$

4.2. How to Predict Position Using Artificial Neural Networks

In this work, three tables were obtained with data from the positions related to the coordinates (x,z) of the robot obtained via Kalman filter. The program to obtain this data was written using the Python language. For each round, 67 pairs of points were provided as output that describe the trajectory from the starting position (0.0) m to close to the goal (7.9, 9.8) m. The training and testing data were divided by the chosen length number 55, that is, there were 55 training data that corresponds to the elements 0 to 54 and 12 test data, which correspond to elements 55 to 66.

The neural network used to train the data obtained via the Kalman Filter consisted of:

- an input layer containing 2 neurons,
- an intermediate layer containing 30 neurons,
- an intermediate layer containing 24 neurons,
- an intermediate layer containing 18 neurons,
- an exit layer containing 2 neurons.

In addition, the loss and optimizer parameters applied were:

- Loss: mean squared_error,
- Optimizer: Adam

5. Results and Discussion

Considering the procedure where it was described how the two Python programs work to estimate the robot's trajectories and predict part of the trajectory by neural networks, from there it was possible to obtain the outputs of both programs in the form of graphs. Figures 3a - 5a show the robot's trajectories in the world containing the twelve objects of different intensities, and the objective is the end of the trajectory. The dotted lines in red show the ideal trajectories where there was no uncertainty and therefore, they are lines straight. The yellow lines show the actual trajectories, where the robot navigated only by odometry, and whether position uncertainty is considered by this method. The dashed lines in green show the trajectories estimated by the Kalman filter considering the fusion of the odometry uncertainties and the cameras in this filter.

In all these three figures it can be observed that there are two NP - new passage. These two NP points show that the robot had two stops, the first of which was to do the triple 60° sweep to try to find the goal. The last point NP is already the goal or close to it.

Figures 3b - 5b show the test data in blue (y_test) which is a part of the position data already estimated by the Kalman filter. The robot's position points predicted by neural networks (y_predict) are shown in red.

The calculation of the percent variance for the x and z coordinates were calculated with Equations 33 e 34:

$$\text{x percent variance} = \frac{(x_{predict} - x_{\_test})}{x_{\_test}} * 100 \tag{33}$$

$$\text{z percent variance} = \frac{(z_{predict} - z_{\_test})}{z_{\_test}} * 100 \tag{34}$$

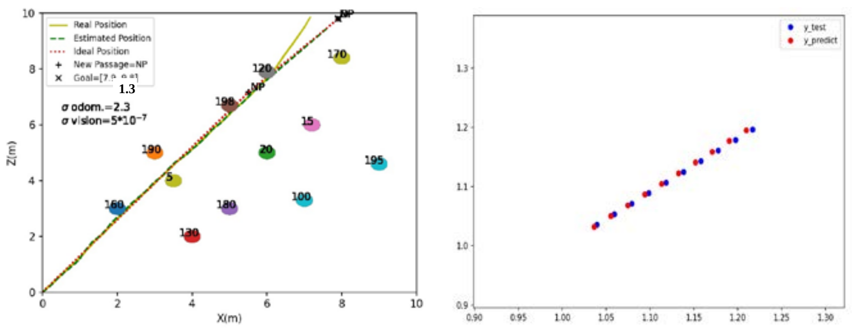

Figure 3. a) Trajectories estimated by the robot: ideal (red), actual (yellow) and estimated (green). b) Part of the positions estimated by the Kalman filter y_test (blue) and positions predicted by the neural networks y_predict (red).

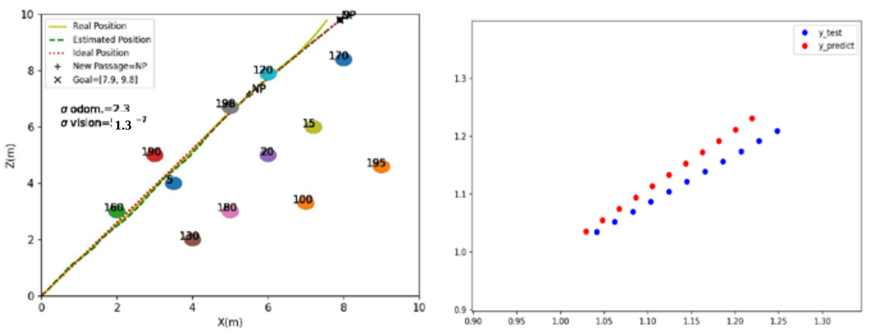

Figure 4. a) Trajectories estimated by the robot: ideal (red), actual (yellow) and estimated (green). b) Part of the positions estimated by the Kalman filter y_test (blue) and positions predicted by the neural networks y_predict (red).

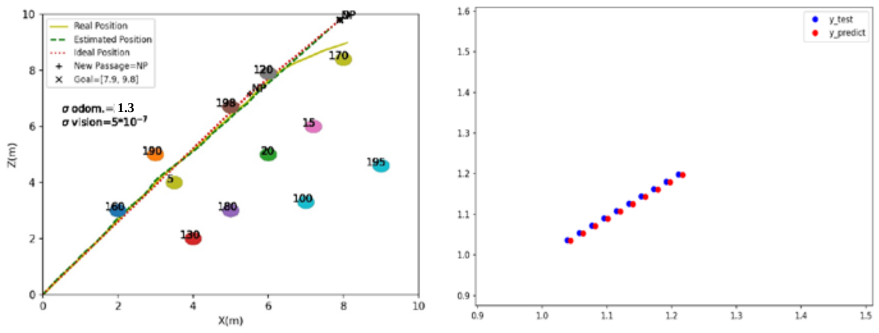

Figure 5. a) Trajectories estimated by the robot: ideal (red), actual (yellow) and estimated (green). b) Part of the positions estimated by the Kalman filter y_test (blue) and positions predicted by the neural networks y_predict (red).

Table 2. Comparison between the positions obtained via Kalman filter (x_test) and via neural networks (x_predict) related to the data in the graphs in Figure 1.

Figure 1	Comparison between the positions obtained via Kalman filter (x_test) and via neural networks (x_predict)		
	x_test (x)	**x_predict (x)**	**x percent variance %**
	1,039461	1,03602	-0,33
	1,059192	1,055289	-0,37
	1,078923	1,074557	-0,4
	1,098654	1,093827	-0,44
	1,118384	1,113096	-0,47
	1,138115	1,132365	-0,51
	1,157846	1,151634	-0,54
	1,177577	1,170903	-0,57
	1,197307	1,190172	-0,6

Figure 1	Comparison between the positions obtained via Kalman filter (x_test) and via neural networks (x_predict)		
	x_test (x)	**x_predict (x)**	**x percent variance %**
	1,217038	1,209441	-0,62

Table 3. Comparison between the positions obtained via Kalman filter (z_test) and via neural networks (z_predict) related to the data in the graphs in Figure 1.

Figure 1	Comparison between the positions obtained via Kalman filter (z_test) and via neural networks (z_predict)		
	z_test	**z_predict**	**z percent variance %**
	1,035735	1,032374	-0,32
	1,053603	1,050485	-0,3
	1,07147	1,068596	-0,27
	1,089338	1,086707	-0,24
	1,107205	1,104818	-0,22
	1,125073	1,122929	-0,19
	1,14294	1,14104	-0,17
	1,160808	1,159151	-0,14
	1,178675	1,177262	-0,12
	1,196543	1,195372	-0,1

Table 4. Comparison between the positions obtained via Kalman filter (x_test) and via neural networks (x_predict) related to the data in the graphs in Figure 2.

Figure 2	Comparison between the positions obtained via Kalman filter (x_test) and via neural networks (x_predict)		
	x_test	**x_predict**	**x percent variance %**
	1,041363	1,028983	-1,19
	1,062044	1,048034	-1,32
	1,082726	1,067085	-1,44
	1,103407	1,086136	-1,57
	1,124088	1,105187	-1,68
	1,14477	1,124238	-1,79
	1,165451	1,143289	-1,9
	1,186132	1,162339	-2,01
	1,206814	1,181365	-2,11
	1,227495	1,200311	-2,21
	1,248177	1,219258	-2,32

Table 5. Comparison between the positions obtained via Kalman filter (z_test) and via neural networks (z_predict) related to the data in the graphs in Figure 2.

Figure 2	Comparison between the positions obtained via Kalman filter (z_test) and via neural networks (z_predict)		
	z_test	**z_predict**	**z percent variance %**
	1,034851	1,035383	0,05
	1,052277	1,054942	0,25
	1,069703	1,0745	0,45
	1,087128	1,094058	0,64
	1,104554	1,113616	0,82
	1,12198	1,133175	1
	1,139405	1,152733	1,17
	1,156831	1,172291	1,34
	1,174256	1,19187	1,5
	1,191682	1,211511	1,66
	1,209108	1,231152	1,82

Table 6. Comparison between the positions obtained via Kalman filter (x_test) and via neural networks (x_predict) related to the data in the graphs in Figure 3.

Figure 3	Comparison between the positions obtained via Kalman filter (x_test) and via neural networks (x_predict)		
	x_test	**x_predict**	**x percent variance %**
	1,038273	1,04368	0,52
	1,057409	1,062937	0,52
	1,076545	1,082194	0,52
	1,095681	1,101451	0,53
	1,114818	1,120708	0,53
	1,133954	1,139964	0,53
	1,15309	1,159222	0,53
	1,172227	1,178478	0,53
	1,191363	1,197736	0,53
	1,210499	1,216993	0,54

Table 7. Comparison between the positions obtained via Kalman filter (z_test) and via neural networks (z_predict) related to the data in the graphs in Figure 3.

Figure 3	Comparison between the positions obtained via Kalman filter (z_test) and via neural networks (z_predict)		
	z_test	**z_predict**	**z percent variance %**
	1,036079	1,034941	-0,11
	1,054118	1,052962	-0,11

Figure 3	Comparison between the positions obtained via Kalman filter (z_test) and via neural networks (z_predict)		
	z_test	**z_predict**	**z percent variance %**
	1,072158	1,070984	-0,11
	1,090197	1,089005	-0,11
	1,108237	1,107027	-0,11
	1,126276	1,125048	-0,11
	1,144316	1,14307	-0,11
	1,162355	1,161091	-0,11
	1,180395	1,179113	-0,11
	1,198434	1,197134	-0,11

6. Conclusion

As can be seen in Figures 1-3a, the trajectories estimated by the robot to its goal, calculated via the Kalman filter, which are shown by the dotted green lines tend to follow the ideal trajectories (in red) for the values of vision uncertainties and odometry used in the simulation. The actual trajectories, in yellow, whose locations are made using odometry, the robot deviated from its goal. This demonstrates that the use of the Kalman filter for the determination from the robot's position, along with the pair of cameras, it was critical to estimating a more accurate trajectory.

Figures 1-3b show the accuracy of predicting a part of the trajectory using neural networks. Tables 2-7 present in more detail the test values, which are the estimated positions of the robot by the Kalman filter (x_test.z_test), differ from the values estimated by the prediction using neural networks (x_predict.z_predict).

Tables 2-3 refer to the position values of Figure 1b, of the trajectories shown in Figure 1a. From Figure 1b, the blue and red dots start closer together at x and they become farther apart at z. Thus, in Table 2, referring to position x, the percent variance values start smaller and then increase and in Table 3, referring to position z, the difference starts larger and then this variation decreases. If the position values estimated by the Kalman filter are compatible with the values of the ideal trajectory, this would mean that in the objective, the robot would be closer to the object in z than in x.

Consider Tables 4-5 in Figure 2b. Figure 1 b shows that the blue and red dots are very far apart in x and z aas x and z increase in value. Tables 5-5 show that the percent variance increases for both coordinates. In this case, it seems that the neural networks were not as accurate to estimate the part of the trajectory as in the case of Figure 1a. The largest percent variance of |2.32|% is still a reasonable value of error.

Analyzing Figure 2c, it can be observed that the estimated and predicted positions always seem to be at the same distance from each other. This means that the percent variance must be close to a constant value. In Tables 6-7 exactly this is observed, where the percent variance in x is around 0.53% and in z around -0.11%. The precision in z is better estimated than in x.

In this work, it can be concluded that the Kalman Filter was essential to determine the positions of the navigating robot in the environment containing the obstacles. Without the filter, the robot deviates from its final goal since the uncertainties of odometry are cumulative. In addition, neural networks with the architecture proposed in this work were able to provide a trajectory for the robot with good accuracy (< | 2.32| %). New machine learning models can be researched in the future to also verify this possibility of estimating the position of the robot in a faster and/or more accurate, since a neural network architecture containing a few layers may require more time to provide the output.

Conflicts of Interest

It has no conflict of interest.

References

[1] Ma, H., Yan L., Xia, Y. and Fu, M. (2020) Kalman Filtering and Information Fusion. 1st Edition, Springer Singapore, 8. https://doi.org/10.1007/978-981-15-0806-6

[2] Diogenes, L. C. M. F. (2008) Uso de Filtro de Kalman e Visão Computacional para a Correção de Incertezas de Navegação de Robos Autonomos. PhD Thesis, Unicamp, Campinas, Brazil.

[3] Piga, N. A., Pattacini, U. and Natale L. (2021) A Differentiable Extended Kalman Filter for Object Tracking Under Sliding Regime. Frontiers in Robotics and AI, 8, 1-20. https://doi.org/10.3389/frobt.2021.686447

[4] Kim S., Petrunin I., Shin, H-S. (2022) A Review of Kalman Filter with Artificial Intelligence Techniques. Proceedings of the 2022 Integrated Communication, Navigation and Surveillance Conference (ICNS), Dulles, USA, 5-7 April 2022, 1-12. https://doi.org/10.1109/ICNS54818.2022.9771520

[5] Pang Z., Wang Y. and Yang F. (2024) Application of Optimized Kalman Filtering in Target Tracking Based on Improved Gray Wolf Algorithm. Scientific Reports, 14, 8955, 1-9. https://doi.org/10.1038/s41598-024-59610-6

[6] Condados, L. G. P. (2020) Controle Embarcado em Robôs com Acionamento Diferencial e Encoders de Baixa Resolução. Bachelor Thesis, Universidade Federal do Rio Grande do Norte, Natal.

[7] Al Azhima S.A.T., Hakim D. L., Nulfatwa R. I., Hakim N. F.A. and Al Qibtiya M. (2024) Cumulative Error Correction of Inertial Navigation Systems Using LIDAR Sensors and Extended Kalman Filter. Indonesian Journal of Electrical Engineering and Computer Science, 34, 2, May 2024, 878-887. http://doi.org/10.11591/ijeecs.v34.i2.pp878-887

[8] Tout B., Chevrie J., Vermeiren L. and Dequidt A. (2022) Design and Tuning of Extended Kalman Filter for Robotic System Identification, 17th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, 2022, 462-467. https://doi.org/10.1109/ICARCV57592.2022.10004282

[9] Abreu, O. S. L., Costa E. A., Silva T. O., Ribeiro M. P., Schnitman L. and Martins, M. A. F. (2022) Implementação em Tempo Real de um Filtro de Kalman Unscented para Estimação de Variáveis de Difícl Medição em um Poço Piloto Operado por BCS. Sociedade Brasileira de Automática (SBA) XXIV Congresso Brasileiro de Automática - CBA 2022, 3, 16-19 October 2022, 4255 - 4260. https://doi.org/10.20906/CBA2022/3753

[10] Li Q., Li R., Ji K. and Dai W. (2015) Kalman Filter and Its Application. 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), Tianjin, China, 2015, 74-77. https://doi.org/10.1109/ICINIS.2015.35

[11] Urrea C. and Agramonte R. (2021) Kalman Filter: Historical Overview and Review of Its Use in Robotics 60 Years after Its Creation, Journal of Sensors, 9674015, 1-21. https://doi.org/10.1155/2021/9674015

[12] Kim J. H., Lee S. H, K., J. G., Jang W. J. and Kim D. H. (2024) Localization of Solar Panel Cleaning Robot Combining Vision Processing and Extended Kalman Filter. Science Progress, 107, 1-29. https://doi.org/10.1177/00368504241250176

[13] Vaz. D. J. F. (2016) Sistema de Navegação para Robot Móvel com Filtros de Kalman. Master Thesis. Academia Militar e Instituto Superior Técnico. Lisboa, Portugal.

[14] Neumann R. (2018) Auto Localização de Robôs Móveis por Fusão de Sensores na Presença de Interferência Eletromagnética. Master Thesis, PUC-Rio, Rio de Janeiro.

[15] Ristic B., Arulampalam S., Gordon N. (2004) Beyond the Kalman Filter. Particle Filters for Tracking Applications, Artech House Publishers. ISBN: 1-58053-631-x.

[16] Gibbs B. P. (2011) Advanced Kalman Filtering, Least-Squares and Modeling. A Practical Handbook-John Wiley & Sons.

[17] Kordic V. (2010) Kalman Filter. InTech, Croatia. https://doi.org/10.5772/233

[18] Martins M. D. L. (2020) Filtragem de Kalman Adaptativa e Robusta para a Reconstrução de Trajetórias. Master Thesis, Universidade da Beira Interior, Portugal.

Author Profile

Luciana C. M. F. Diogenes was born in the city of São Paulo, Brazil in 1980. She earned a bachelor's degree in physics at Unicamp (University of Campinas, Brazil) in 2001, master's in physics at Unicamp in 2004 and Ph.D. in Mechanical Engineering at Unicamp in 2008, where a year of the doctorate was held at the Faculty of Engineering of Porto in Portugal. She completed an MBA in Project Management in 2014 at FGV - Fundação Getúlio Vargas, Brazil. She has worked as a project specialist in research and development in ThyssenKrupp at Campo Limpo Paulista, Brazil for 5.5 years. She also served as a university professor at UEMG - Minas Gerais State University - in the cities of Frutal and Ituiutaba, both in Brazil. In UEMG Frutal, she acted as research coordinator for a one-year term. Currently, the author is dedicated to works as Python Developer and is open to new cooperations.