Extended Kalman Filter Enhanced by Neural Network to Solve the SLAM Problem

Bruno França Coelho
Programa de Pós graduação em Engenharia Elétrica
Universidade Federal do Maranhão
Maranhão, Brasil
bruno.fc@discente.ufma.br

João Viana da Fonseca Neto
Programa de Pós graduação em Engenharia Elétrica
Universidade Federal do Maranhão
Maranhão, Brasil
joao.fonseca@ufma.br

***Abstract*—This work presents a way for online estimation of the location and mapping of a non-holonomic robot by means of an algorithm that uses EKF and in the output of this algorithm, a multilayer perceptron neural network (MLP) has been added that aims to improve the estimation of the robot pose in an unfamiliar environment. The effectiveness was proven through the comparison between the EKF-SLAM and the EKFMLP-SLAM, where it was evidenced a significant improvement in relation to the location of the poses of the robot.**

***Index Terms*—EKF, SLAM, MLP, artificial neural network**

I. Introduction

The ability to place a robot in an unknown location and environment and then have it build a map, using only relative observations of the environment, and then use this map simultaneously to navigate, actually turns that robot into "aautonomous" [1].

The problem of simultaneous location and mapping, also known as SLAM, has attracted immense attention in the mobile robotics literature. SLAM addresses the problem of building a map of an unknown environment from a sequence of measurements of landmarks obtained from a moving robot. As the movement of the robot is also subject to errors, the mapping problem necessarily induces a problem of location of the robot - hence the name SLAM. SLAM is considered by many to be an essential capability for autonomous robots operating in environments where accurate maps and positioning are not available [2]. Thus, the main advantage of SLAM is that it eliminates the need for artificial infrastructures or a priori topological knowledge of the environment.

A SLAM algorithm consists of exploring the unknown environment to build or update your map and determine the position of the robot on the map based on a series of actions and observations [3]. There are many applications of the SLAM solution, such as autonomous vehicles [4], [5], minimally invasive surgery [6], [7] and harvest [8]. SLAM is a complex problem due to the strict requirements in mobile robots, especially related to the robustness, computational efficiency and precision of the [9] algorithm. Although the position of the robot can be obtained through signals from the Global Positioning System (GPS), they have limitations, so they can be blocked by buildings or thick clouds. The SLAM method can be used to resolve the limitations of this GPS signal.

Nonlinear SLAM is predominantly implemented as an extended Kalman filter (EKF), where system noise is assumed to be Gaussian and nonlinear models are linearized to suit the Kalman filter algorithm. EKF-SLAM represents the state uncertainty by an approximate mean and variance. This is a problem in two ways. First, these moments are approximated due to linearization and may not accurately correspond to the first and second "true" moments. Second, the true probability distribution is non-Gaussian, so that even the true mean and variance may not be an adequate description. These factors affect how the SLAM probability distribution is projected over time, over a sequence of movements and measurements, and how the approximation errors accumulate [10].

In this work we present an algorithm for online estimation of the location of a non-holonomic robot by means of an algorithm that uses EKF and at the output of this algorithm is added a multilayer perceptron neural network (MLP) [11]-[15], which aims to improve the estimation of the pose of the robot in an unknown environment.

This paper is organized as follows: Section 2 presents the Simultaneous Location and Mapping of a robot using the Extended Kalman Filter and a mathematical modeling based on the speed of a non-holonomic robot; Section 3 presents an MLP neural network with two hidden layers to improve the estimation of the robot's pose in relation to the unknown environment; Section 4 presents the results and computational experiments related to the EKF-SLAM and EKF-SLAM algorithm with the addition of an MLP neural network. Next, the conclusions about the work developed are exposed and finally the references used for the elaboration of this research are presented.

II. EKF applied in SLAM

The Kalman Filter (KF) is a recursive filter proposed by Kalman [16], and is the most studied implementation of the Bayes Filter. It estimates the current state of a linear dynamic system from the observations acquired so far.

In KF, the belief is represented by a multivariable Gaussian distribution, parameterized by an average μ_t and covariance Σ_t. The average μ_t is a vector that has the same dimension as

x, while the covariance Σ is a positive, semi-defined square matrix, with the same dimensionality as x. An important feature is that Gaussian distributions are unimodal, that is, they have a single maximum, making it possible to use them for tracking the pose in many robotic problems [9].

The linear dynamic model in (1) describes the state prediction used in KF, and (2) measurement prediction:

$$x_t = A_t x_{t-1} + B u_{t-1} + \varepsilon_x, \qquad \varepsilon_x \sim \mathcal{N}(0, R) \tag{1}$$

$$\hat{z}_t = C_t x_t + \varepsilon_z, \qquad \varepsilon_z \sim \mathcal{N}(0, Q) \tag{2}$$

being, $x_t \in \Re^{n\times 1}$ e $x_{t-1} \in \Re^{n\times 1}$ the state vectors, $u_{t-1} \in \Re^{m\times 1}$ the control vector and $\hat{z}_t \in \Re^{k\times 1}$ the measurement vector. A_t and B_t are matrices $\Re^{n\times n}$ and $\Re^{n\times m}$, respectively, and describe the temporal evolution of the state. The random Gaussian vectors $\varepsilon_x \in \Re^{n\times 1}$ and $\varepsilon_x \in \Re^{k\times 1}$, have zero mean and covariance denoted by R e Q, respectively, modeling the uncertainties introduced by state transition and measurement. The matrix $C_t \in \Re^{k\times n}$ performs the mapping between x_t and $\hat{z}_t$, in which k is the dimension of the measurement vector $\hat{z}_t$ and $\mathcal{N}$ represents a random noise component with (approximately) zero mean [9].

Algorithm 1 EXTENDED KALMAN FILTER

1: **Inputs:** $\mu_{t-1}, \Sigma_{t-1}, u_{t-1}, z_t$
2: Prediction:
3: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
4: $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
5: **Update:**
6: $K = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
7: $\mu = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
8: $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
9: **return** $\mu_t, \quad \Sigma_t$

The entry of EKF is the belief in time $t-1$, represented by μ_{t-1} e Σ_{t-1}. To update these parameters, use the control input u_{t-1} and measurement z_t. The output of the EKF is the belief in time t. The variable K_t represents Kalman gain, and specifies the degree to which the measure z_t is incorporated into the new state estimate. In line 7, the average is manipulated $\bar{\mu}_t$, adjusting it proportionally to the gain K_t and the deviation from the current measure z_t. Finally, in line 8, the new covariance of the belief is calculated. Process and measurement noises are assumed to be independent and normally distributed with covariance R_t and Q_t, respectively. EKF is presented in the Algorithm 1.

The assumptions of linearity of the observability and transience functions are crucial for the use of KF. However, these functions are rarely linear in practice. The Extended Kalman Filter (EKF) is used when the system has non-linearity. EKF linearizes the model around the current estimation and then uses the traditional FK equations. The EKF is based on the linearization of the state transition function and the measurement function by means of the Taylor series approximation, tracing the first order term, and disregarding higher order terms. This approximation can cause problems if the nonlinearities are very large, causing the algorithm to diverge.

Observing the Algorithm 1, you can see that (1) and (2) were relaxed, and the state transition probability and measurement are described by $x_t = g(u_t - 1, x_{t-1}) + \varepsilon_x$ and $\hat{z}_t = h(xt) + \varepsilon_z$, respectively, for functions g and h arbitrary. The idea of EKF is to linearize g and h using the first-order Taylor expansion approach.

A. Kinematic model of the mobile robot

The model of the current robot pose $\mu_t = (x_t, y_t, \theta_t)^T$ after executing the motion command $u_{t-1} = (v_{t-1}, \omega_{t-1})^T$ in the state $m_{t-1} = (x_{t-1}; y_{t-1}; \theta_{t-1})^T$ is given by:

$$\underbrace{\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}}_{\mu_t} = \underbrace{\begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \Delta t \cos(\theta_{t-1})\, v_{t-1} \\ \Delta t \sin(\theta_{t-1})\, v_{t-1} \\ \Delta t \omega_{t-1} \end{bmatrix}}_{g(u_{t-1}, \mu_{t-1})} + \underbrace{\mathcal{N}(0, R)}_{\varepsilon_x}, \tag{3}$$

being Δt the sampling time, v_{t-1} the velocity linear , ω_{t-1} the angular velocity, and μ_t is the estimation of the robot's pose. It can be observed in (3) that $g(u_{t-1}, \mu_{t-1})$ it is clearly non-linear, so the EKF should be used to estimate the pose. The matrix G_t and H_t in Algorithm 2 are given by:

$$G_t = \frac{\partial g(u_{t-1}, \mu_{t-1})}{\partial \mu_{t-1}} = \begin{bmatrix} 1 & 0 & -\Delta t sin(\theta_{t-1}) \\ 0 & 1 & \Delta t sin(\theta_{t-1}) \\ 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial \bar{\mu}_t} = -\frac{1}{q} \begin{bmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & \sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -1 & -\delta_y & \delta_x \end{bmatrix}, \tag{5}$$

being $\delta = \begin{pmatrix} m_x - \mu_x & m_y - \mu_y \end{pmatrix}^T$, $q = \delta^T \delta$ and m represents the coordinate of the landmark.

B. EKF-SLAM Algorithm

The Algorithm 2 presents an application of the extended Kalman filter to solve the SLAM problem.

For a better understanding, the following are presented the inputs and outputs of the algorithm for SLAM using an EKF:

Inputs

- The average of the pose estimate in the previous time step:

$$\mu_{t-1} = (x_{t-1}\ y_{t-1}\ \theta_{t-1}\ m_{1,x}\ m_{1,y}\ m_{2,x}\ m_{2,y} \cdots m_{n,x}\ m_{n,y})^T.$$

  - x_{t-1} is the robot's previous x position in meters;
  - y_{t-1} is the robot's previous y position in meters;
  - θ_{t-1} is the robot's previous heading in radians;
  - $m_{i,x}$ is the x position of the landmark i in meters;
  - $m_{i,y}$ is the y position of the landmark i in meters.
- F_x is a matrix that maps the three-dimensional state vector to a dimension vector $(5 \times 2n + 3)$, being n the number of landmarks.
- The covariance matrix from the previous timestep: Σ_{t-1}
  - represents the uncertainty of the pose estimate.

- The control inputs in the current time step: $u_t = (v_t\, \omega_t)$
  - v_t s the robot's linear velocity in meters/sec;
  - ω_t is the robot's angular velocity in radians/sec.
- Current measurements: $z_t = \left(z_t^1\ z_t^2\ \dots\ z_t^k\right)$, k represents the number of measurements over time t
  - each $z_t^i = \left(r_t^i\ \phi_t^i\right)^T$ where
    - r_t^i is the range to landmark i in meters;
    - ϕ_t^i is the bearing to landmark i in radians.

Outputs

- the updated pose estimate, μ_t;
- updating the covariance matrix, Σ_t.

Algorithm 2 FKE-SLAM

Initialize $N_t = N_{t-1}$, F_x, G_t e Q_t
$\bar{\mu}_t = g(u_t, \mu_{t-1})$
$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + F_x^T R_t F_x$
for all $z_t^i = \begin{bmatrix} r_t^i & \phi_t^i \end{bmatrix}^T$ **do**

$$\begin{bmatrix} \bar{\mu}_{N_{t+1},x} \\ \bar{\mu}_{N_{t+1},y} \end{bmatrix} = \begin{bmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \end{bmatrix} + r_t^i \begin{bmatrix} \cos\left(\phi_t^i + \bar{\mu}_{t,\theta}\right) \\ \sin\left(\phi_t^i + \bar{\mu}_{t,\theta}\right) \end{bmatrix}$$

  for $k = 1$ **to** N_{t+1} **do**
    $z_i = h(x_t, j, m) + \mathcal{N}(0, Q_t)$
    $F_{x,j} = \begin{bmatrix} I^{3\times 3} & 0 & 0 & 0 \\ 0 & \underbrace{0}_{2j-2} & I^{2\times 2} & \underbrace{0}_{2j-2} \end{bmatrix}$
    ${H_t}^{k\ low} = \frac{\partial h(\bar{\mu}_t)}{\partial \bar{\mu}_t}$
    $H_t^k = {H_t}^{k\ low} F_{x,k}$
    $\Psi_k = H_t^k \bar{\Sigma}_t \left[H_t^k\right]^T + Q_t$
    $\pi_k = ({z_t}^i - \hat{z}_t^k) {\Psi_k}^{-1} ({z_t}^i - \hat{z}_t^k)$
  end for
  $\pi_{N_t+1} = \alpha$
  $j(i) = \arg\min \pi_k$
  $N_t = \max\{N_t, j(i)\}$
  ${K_t}^i = \bar{\Sigma}_t \left[H_t^{j(i)}\right]^T {\Psi_{j(i)}}^{-1}$
end for
$\mu_t = \bar{\mu}_t + \Sigma_i K_t^i (z_t^i - \hat{z}_t^{j(i)})$
$\Sigma_t = (I - \Sigma_i K_t^i H_t^{j(i)}) \bar{\Sigma}_t$
return $\mu_t,\quad \Sigma_t$

Starting the iterative process, there is the observation and prediction step, which consists of estimating the current state of the robot, then performing the calculation of the Jacobian matrices H_t, G_t and then calculate the covariance matrix $\bar{\Sigma}_t$. Continuing, another iterative loop responsible for making the updates begins. Then the distances between the landmarks and the current position of the robot are calculated and then it is possible to calculate the error between the estimated and the observed positions. Kalman's gain is also calculated at this stage K_t, updating the estimated state $\hat{\mu}_t$ and updating the covariance matrix error.

III. MLP Neural Network

In order to improve the estimation of the robot's location in relation to the environment, a multilayer perceptron neural network (MLP) was added. The training of the MLP network is given by the algorithm *backpropagation* which is based on the calculation of the error in the output layer of the neural network, the values of the weights of the vector w are recalculated, performing the back propagation until reaching the input layer of the network [17]. The general formula for updating weights is given by

$$w = w - \eta \frac{\partial E}{\partial w}, \tag{6}$$

being η represents the learning rate of the neural network, the weight value in the current iteration will be the weight value in the previous iteration. The negative sign is that the algorithm is going in the opposite direction to the gradient. The error function is defined by

$$E(y, \hat{y}) = \sum_{N}^{i=1} (y_i - \hat{y}_i)^2. \tag{7}$$

The MLP used in this work has two layers and the hyperbolic tangent was chosen as the activation function. The input signals in the neuron of the first layer is the estimated pose of the robot through the Algorithm 2 .

$$\mathbf{x} = (x \quad y \quad \theta)^T, \tag{8}$$

upon reaching the neuron, they are multiplied by their respective synaptic weights w, generating the activation potential z that is given by

$$z = \sum_{i=1}^{N} x_i w_i + b, \tag{9}$$

being b the bias, the value z goes through an activation function σ.

The following are the calculations for obtaining the formula for updating the weights of the backpropagation algorithm. In MLP, the $\mathbf{x}$ vector passes through the initial layer whose output values are linked to the inputs of the next layer, until the network provides the output values of the last layer as a result.

The calculation of the error derivative in relation to the z activation potential of layer (2), using the chain rule

$$\frac{\partial E}{\partial z_i^{(2)}} = \frac{\partial E}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i^{(2)}} = -2\,(y_i - \hat{y}_i)\,\sigma'\left(z_i^{(2)}\right), \tag{10}$$

to simplify the equations we indicate that

$$\frac{\partial E}{\partial z_i^{(2)}} = \delta_i^{(2)}. \tag{11}$$

Using the chain rule again, the partial derivative of the error is computed as a function of the weight $w[j,i]$

$$\frac{\partial E}{\partial w_{j,i}^{(2)}} = \frac{\partial E}{\partial z_i^{(2)}} \frac{\partial z_i^{(2)}}{\partial w_{j,i}^{(2)}} = \delta_i^{(2)} \frac{\partial}{\partial w_{j,i}^{(2)}} \left(\sum_{k=1}^{M} w_{k,i}^{(2)} y_k^{(l)} + b_i^{(2)} \right) = \delta_i^{(2)} y_j^{(l)}. \tag{12}$$

The calculation in relation to the bias is similar, resulting in

$$\frac{\partial E}{\partial b^{(2)}} = \delta_i^{(2)} \tag{13}$$

Replacing (12) em (6), we have

$$w_{j,i}^{(2)} = w_{j,i}^{(2)} - \eta \delta_i^{(2)} y_j^{(1)}, \tag{14}$$

likewise the bias update is given by

$$b_i^{(2)} = b_i^{(2)} - \eta \delta_i^{(2)}. \tag{15}$$

In the same way, calculations are performed for the anterior neuron. Then it is necessary to calculate the partial derivative in relation to the y_i output of layer (1), resulting in

$$\frac{\partial E}{\partial y_i^{(1)}} = \sum_{j=1}^{N} \frac{\partial E}{\partial z_j^{(2)}} \frac{\partial z_j^{(2)}}{\partial y_i^{(1)}} = \sum_{j=1}^{N} \delta_j^{(2)} w_{i,j}^{(2)}. \tag{16}$$

continuing we have:

$$\frac{\partial E}{\partial z_i^{(1)}} = \frac{\partial E}{\partial y_i^{(I)}} \frac{\partial y_i^{(I)}}{\partial z_i^{(I)}} = \left(\sum_{j=1}^{N} \delta_j^{(2)} w_{i,j}^{(2)} \right) \sigma' \left(z_i^{(I)} \right) = \delta_i^{(I)}, \tag{17}$$

being δ the local gradient in relation to the i-th layer neuron (1). So it is possible to calculate

$$\frac{\partial E}{\partial w_{j,i}^{(1)}} = \frac{\partial E}{\partial z_i^{(1)}} \frac{\partial z_i^{(I)}}{\partial w_{j,i}^{(I)}} = \delta_i^{(I)} \frac{\partial}{\partial w_{j,i}^{(l)}} \left(\sum_{k=1}^{L} w_{k,i}^{(I)} x_k + b_i^{(I)} \right) = \delta_i^{(I)} x_j. \tag{18}$$

Replacing (6) in (18), it has

$$w_{j,i}^{(I)} \leftarrow w_{j,i}^{(1)} - \eta \delta_i^{(1)} \mathbf{x}_j. \tag{19}$$

Next, the computational experiments and the results obtained are presented.

IV. Computational Experiments

Simultaneous location and mapping (SLAM) is a key component for mobile robot navigation that allows for many applications in service robots. The ability to acquire accurate maps of an environment is very important for robots to perform various tasks with a high degree of autonomy. This section presents the initial results of implementing the algorithms for locating and mapping robots $EFK - SLAM$.

The simulations were performed on a computer that has a Core i5-8300H 2.30 GHz processor, and 16 GB of RAM, the codes were executed in the MATLAB program.

Inputs are initialized (μ_{t-1} Σ_{t-1} u_t z_t) on algorithm $EFK - SLAM$, for this simulation it was defined $\mu_{t-1} = (0\ 0\ 0)^T$, $\Sigma_{t-1} = I_{3\times3}$, being I an identity matrix, both control entries u_t and observations z_t were subjected to noise with Gaussian distribution, the control input u_t is shown in Fig.1.

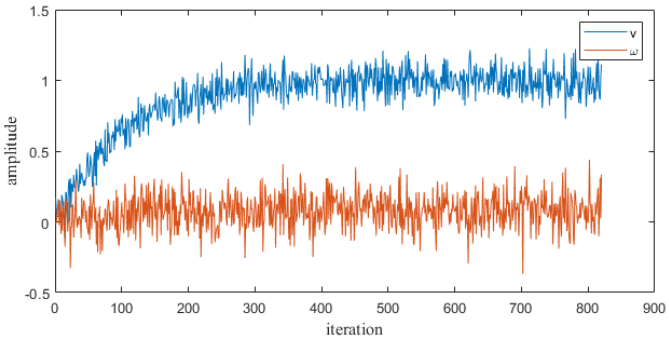

Fig. 1. Inputs $u = (v\,\omega)^T$ of the robot with addition of Gaussian noise.

Then the reference points or (landmarks) are added that will be mapped by the robot, in total there were 63 points, these are spatial coordinates in the xy plane. The Observation Range was determined to be 20 meters, that is, the limit at which the robot can identify the landmarks.

A circular path robot's was defined through the control input u_t, it started from position $(0, 0)$ and executed a circular path as can be seen in Fig. 2, the blue line represents the actual displacement of the robot, the red line corresponds to the estimation of the robot's location through the FKE filter, the black line is the Dead reckoning, that is, estimate of the robot's position without the when using filtering, the circles are landmarks and the blue asterisks represent the estimate of the position of the landmarks.

In Fig.2 the simulation is illustrated when the robot moves from its initial pose, it has its own pose uncertainty, detects the nearby landmarks that are already mapped with its pose uncertainty and obtains a measurement uncertainty from range sensor. The final aspect of the simulation when the robot detects the first reference point again, the uncertainty of its current pose is reduced. This observation also reduces the uncertainty for the other landmarks on the map that were previously seen by the same robot.

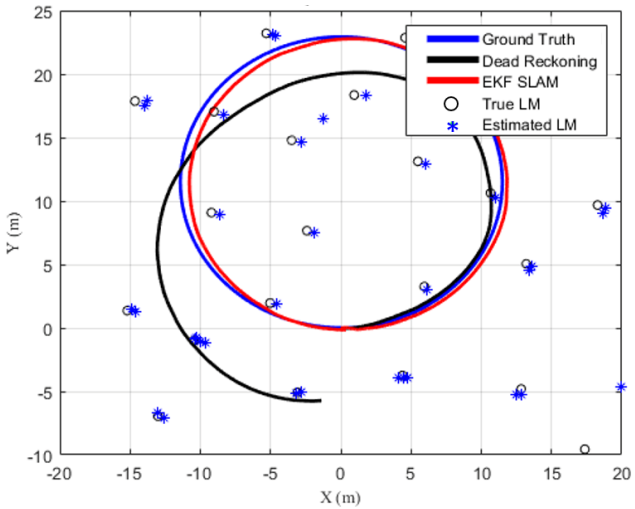

Fig. 2. Estimated robot path and waypoint positions using $EKF - SLAM$, disturbed by Gaussian noise, with a range of 20 meters.

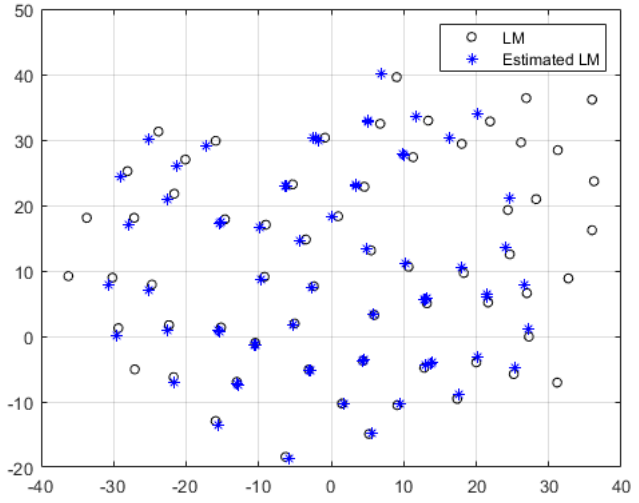

Fig. 3. Map generated using the $EFK - SLAM$ algorithm, with a range of 20 meters.

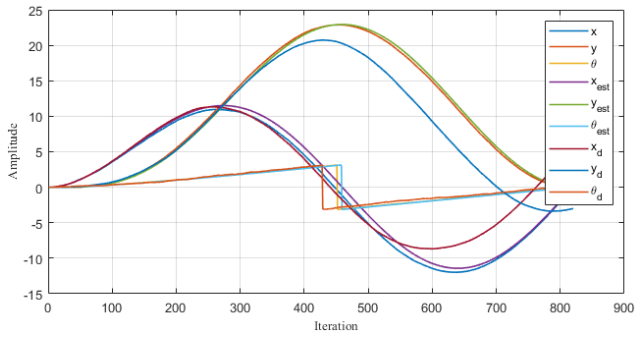

Fig. 4. State estimation, with a range of 20 meters.

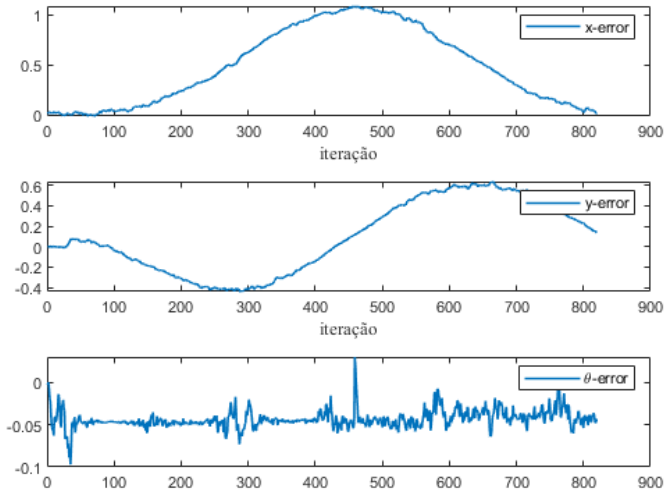

Fig. 5. Error estimating the poses of the mobile robot (EKF-SLAM algorithm), x, y and θ. The horizontal axis represents the number of iterations and the vertical the signal amplitude.

The map consists of a fixed number of landmarks. The map generated after the robot passes through an unknown environment and makes a circular path is shown in Fig. 3, the circles represent the marks and asterisks in blue are the estimates of the positions of the marks. Where it is possible to observe that the points that have not been mapped are out of the range of the sensor and the points that have more than one asterisk are the points at which the sensor has captured the reference point more than once.

The estimation of the robot states, with x and y being the dimensional coordinates θ the orientation. The robot pose is described by the vector $(x\ y\ \theta)^T$, the subscript (est) are the states estimated by means of the EKF-SLAM filter, the subscript (d) are the states estimated by Dead reckoning, these poses are illustrated in Fig. 4.

The error related to the estimation of the robot pose is shown in Fig. 5, this error is the difference between the actual poses and those estimated using EKF.

The MLP neural network was added to the EKF-SLAM algorithm to improve the accuracy of the robot's pose estimation, Fig. 6 shows a significant improvement in the location of the mobile robot. Where it is possible to see that the accuracy of the robot's location has been greatly improved and that it was only possible to notice the difference between the actual and estimated position of the robot through the enlargement of a section of the image.

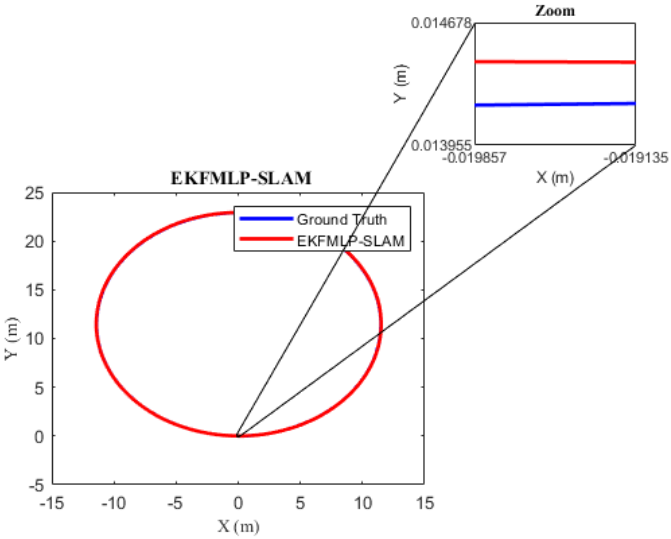

Fig. 6. Location through EFKMLP-SLAM.

Using the EKFMLP Algorithm, it is possible to observe that a better precision in the estimation of the robot's location was obtained, however the algorithm does not always converge. The Fig. 7 graphically shows the curve of the neural network loss function, where it is possible to notice that in approximately 400 iterations this function has been minimized.

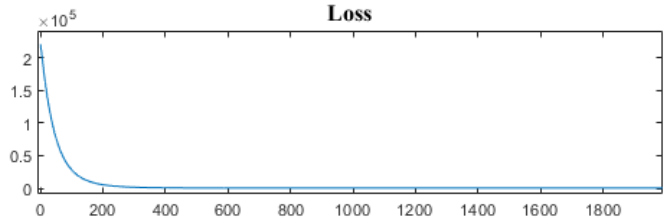

Fig. 7. Loss function.

It is also possible to observe this improvement in the estimation through the error graph.

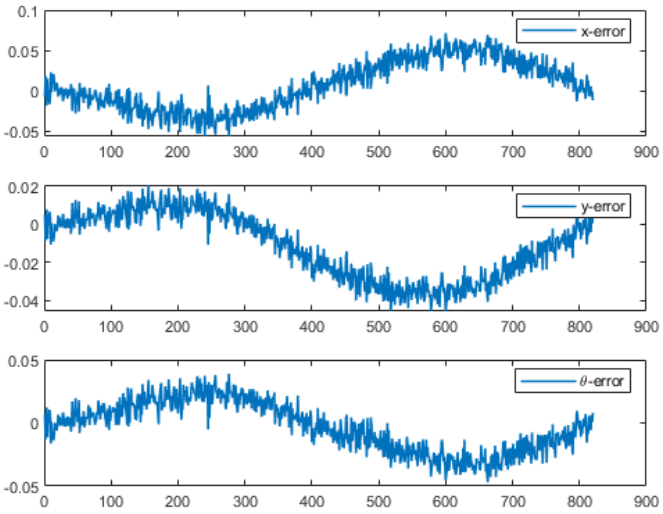

Fig. 8. Error estimating the poses of the mobile robot (Algorithm EKFMLP-SLAM), x, y and θ. The horizontal axis represents the number of iterations and the vertical the signal amplitude.

CONCLUSIONS

The extended Kalman filter proved to be efficient for solving the SLAM problem of a nonhomonic planar robot in an unknown environment, it has some limitations when many benchmarks are added. Simulations were shown in which it was possible to observe the accuracy of the mapping and location of a robot. An MLP neural network was also added to the output of the EKF-SLAM algorithm in order to improve the estimation of the robot's location and thereby improve the mapping performed by the algorithm. The EKFMLP-SLAM proved to be an efficient technique in estimating the location of the robot, presenting a better result in relation to the EKF-SLAM.

ACKNOWLEDGMENT

The authors are indebted to the Federal University of Maranhão (UFMA), the Graduate Program in Electrical Engineering (PPGEE), the State University of Maranho (UEMA), the Embedded Systems and Intelligent Control Laboratory (LABSECI), the Coordination of Undergraduate Personnel Improvement (CAPES), and the National Council for Scientific and Technological Development (CNPq) for the development infrastructure and financial support.

REFERENCES

[1] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," in IEEE Transactions on Robotics and Automation, vol. 17, no. 3, pp. 229-241, June 2001.

[2] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using FastSLAM," 2003 IEEE International Conference on Robotics and Automation, 2003, vol.2, pp. 1985–1991 .

[3] H. Sobreira, C.M. Costa ,I. Sousa, et al. "Map-Matching Algorithms for Robot Self-Localization: A Comparison Between Perfect Match, Iterative Closest Point and Normal Distributions Transform". J Intell Robot Syst, 2019, vol. 93, pp. 533-–546.

[4] S. Kato et al., "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems," 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), 2018, pp. 287–296.

[5] J. Tang , S. Liu, J. Gaudiot, S. Wu, L. Li, "Creating Autonomous Vehicle Systems". Morgan and Claypool Publishers, UK, 2020.

[6] J. Song, J. Wang, L. Zhao, S. Huang and G. Dissanayake, "MIS-SLAM: Real-Time Large-Scale Dense Deformable SLAM System in Minimal Invasive Surgery Based on Heterogeneous Computing," in IEEE Robotics and Automation Letters, Oct. 2018, vol. 3, no. 4, pp. 4068–4075.

[7] P. Mountney and G. Yang, "Dynamic view expansion for minimally invasive surgery using simultaneous localization and mapping," 2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2009, pp. 1184–1187

[8] F. Cheein and G. Steiner and G. Paina and R. CarelliG. Eason, "Optimized EIF-SLAM algorithm for precision agriculture mapping based on stems detection", Computers and Electronics in Agriculture, vol. 78, pp. 195–207, 2011.

[9] S. Thrun, W. Burgard, D. Fox and R.C. Arkin, "Probabilistic Robotics", Intelligent Robotics and Autonomous Agents series, MIT Press, 2005.

[10] T. Bailey, J. Nieto, J. Guivant, M. Stevens and E. Nebot, "Consistency of the EKF-SLAM Algorithm," 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp. 3562-3568.

[11] F. S. Vidal, P. F. F. Rosa, A. d. M. Neto and T. E. A. d. Oliveira, "Multilayer Perceptron Use in a Mapping Task by Cooperating Robots," 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence, 2013, pp. 580-585.

[12] J. Jung, S. Lee and H. Myung, "Indoor Mobile Robot Localization and Mapping Based on Ambient Magnetic Fields and Aiding Radio Sources," in IEEE Transactions on Instrumentation and Measurement, vol. 64, no. 7, pp. 1922-1934, July 2015.

[13] Z. Alsayed, G. Bresson, A. Verroust-Blondet and F. Nashashibi, "2D SLAM Correction Prediction in Large Scale Urban Environments," 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 5167-5174.

[14] F. S. Vidal, P. F. F. Rosa, A. de Melo Neto and T. E. A. de Oliveira, "Cooperating robots for mapping tasks with a multilayer perceptron," IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society, 2013, pp. 4073-4078.

[15] A. Raibolt, A. Angonese and P. Rosa, "Comparative Evaluation of Feature Descriptors Through Bag of Visual Features with Multilayer Perceptron on Embedded GPU System," 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), 2020, pp. 1-6.

[16] R. E Kalman, "A New Approach to Linear Filtering and Prediction Problems." ASME. J. Basic Eng. March 1960; 82(1): pp.35—45.

[17] S. Haykin, "Neural networks and learning machines", Prentice Hall, 2009.