Neural Network Regularization of an Inertial Odometry Estimation for Position Control of a Mobile Robot

Marcus V. P. Lima
system integrated department
University of Campinas
Campinas, Brazil
marcuslima3@gmail.com

Pedro R. M. Silva
system integrated department
University of Campinas
Campinas, Brazil
engpedroramon@gmail.com

César H. C. Quiroz
system integrated department
University of Campinas
Campinas, Brazil
cesar@fem.unicamp.br

Paulo R. G. Kurka
system integrated department
University of Campinas
Campinas, Brazil
kurka@fem.unicamp.br

***Abstract*—Localization is one of the main tasks of autonomous mobile robots. There are many approaches on how to determine the robot's position with reasonable precision, as with the use of sensor fusion (IMU, GPS, Image, LiDAR). Even though it is possible to achieve high precision with these sensors combined, a solution that requires less resources in terms of processing, energy consumption and yet provide a good enough estimation for position control is sought. This paper presents the application of an artificial neural network to improve position estimates from inertial measurement in a mobile robot navigating in an indoor environment.**

***Keywords*— neural networks, inertial measurement units, mobile robots, position control, autonomous robots**

I. Introduction

Precise localization is essential for autonomous robots. Many applications such as autonomous urban vehicles [5], drones [16] and underwater vehicles [9] require precise localization in order to follow a desired path. Generally, precision comes with a cost: high quantity of data, or as referred in literature, big data [2]. This is a result of the combination of a network of sensors at a high sampling rate, such as LiDAR's, GPS, Inertial Measurement Unit's (IMU's) and image processing. The handling of such large quantity of information requires powerful processing units and, consequently, high energy consumption. If the subject is set to mobile robots, those issues are critical, since these vehicles are embedded with relatively low processing power and need as much energy autonomy as possible.

One important task of a mobile robot might be to explore remote or unknown areas. In such case, it would be highly efficient if the robot could explore the environment autonomously and return to a determined location precisely and safely. Even though this is possible using the sensor network mentioned, one has to consider rough environmental conditions, such as GPS signal lost (underground, indoor), non-reflective or low reflective surfaces interfering with the LiDAR, dark places such as caves and deep underwater making the image processing unreliable.

A set of sensors that can perform well in most general situations is the IMU. However, this sensor unit suffers from accumulated error, including Abbe error [18], leading to an ever-increasing difference between the actual and the estimated location. A constant error in attitude rate, results in a quadratic error in velocity and cubic error growth in position [15]. To this matter, estimating the robot's position with IMU-only requires additional procedures in order to obtain a reasonable precision for position control.

A. Related Work

On the subject of inertial odometry, [13] integrates an IMU to a stereo camera vision serving as a proprioceptor sensor for a SLAM algorithm using Kalman Filter. The IMU increases the robustness of the pose estimation via visual odometry, due to the motion information added to the algorithm by the inertial sensors.

[4] proposes a foot-mounted IMU for pedestrian tracking. Its algorithm detects each step of the pedestrian, using acceleration measurements to identify the motion being performed (the gait cycle), making it easier to integrate the acceleration at the right intervals of movement.

From the neural network application perspective, [3] describes a method using ultrasonic sensors and a neural network to correct odometry errors. This work is a good starting point for neural network application in odometry correction.

The work of [17] presents a technique of the odometry error calibration both indoor and outdoor using a back-propagation and a feed-forward model. The experiment is done with a differential robot using a laser scanner with a computer to process the data. The results are a convenient reference for neural network modelling and error estimation.

Another approach using neural network aided improvement is shown in [20]. Their work addresses a method for improving accuracy of a Neural Network aided Extended Kalman Filter (EKF) by compensating for and odometry error of a robot. The

The authors would like to thank the University of Campinas, CNPq and FAPEAM

results validate this method as an interesting post processing possibility.

B. *Objectives and Limitations*

The objective of this paper is to implement and evaluate a methodology of position estimation via IMU, combined with a neural network regularization to improve the estimation results in order to allow a more precise position control of the robot.

An experiment is done in an indoor environment with a differential mobile robot performing a trajectory composed of curves and lines in succession. The reference and actual position of the robot are measured with a single camera positioned on the top of the room, parallel to the floor, using the software Kinovea to track the robot's movement. Kinovea is a free software application for the analysis, comparison, measurements and evaluation, especially suitable for biomechanical analyzes. However, it is necessary a literature about Kinovea to ensure the reliability and validity of the results.

The work of [19] purpose a study to investigate the intra-rater and inter-rater reliabilities of the Kinovea software program for the measurement of dominant wrist joint range of motion in healthy participants. According to [19] the results obtained of the Kinovea software tracking system is highly reliable in both inter and intra-rater assessment o the wrist joint ROM.

In this work, the camera is used to record the movement of the robot. After that, the software Kinovea was used to processing of the video and insert a reference on the robot to tracking during the path. The estimations consider a 2-D displacement only.

This paper is organized as follows. First, the methodology of position estimation with the IMU is explained, followed by additional data handling and filtering. The neural network design, training and implementation is explained next, followed by additional signal filtering. Finally, the results evaluating the performance of the neural network estimation are presented and discussed.

II. Methodology

A. *Odometry*

The MPU 6515 sensor inbuilt in a Nexus 5 smartphone, which was attached on the robot, was used is used as platform in this research to obtain the acceleration and orientation signals and data from inertial sensors.

The output data from accelerometer can be demonstrated in Eq. 1:

$$a = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = R \cdot (g - a_r) \tag{1}$$

where a $[m/s]^2$ is accelerometer output data with X, Y and Z axes parts, a_r is the linear acceleration measured in the earth's reference frame r, R is the orientation matrix, and g $[m/s]^2$ is gravitational force. In Earth's gravitational field, gravitation is presented by -1g or +1g value in one of axes [6].

The orientation of the smartphone can be defined by its roll, pitch and yaw rotations from an initial [14]. The rotation matrices, which transform a vector under a rotation of the coordinate system by angles ϕ for roll, θ for pitch and ψ for yaw about the x, y and z axes respectively, are showed in Equations 2, 3 and 4:

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\Delta\phi(l)) & \sin(\Delta\phi(l)) \\ 0 & -\sin(\Delta\phi(l)) & \cos(\Delta\phi(l)) \end{pmatrix} \tag{2}$$

$$R_y(\theta) = \begin{pmatrix} \cos(\Delta\theta(l)) & 0 & -\sin(\Delta\theta(l)) \\ 0 & 1 & 0 \\ \sin(\Delta\theta(l)) & 0 & \cos(\Delta\theta(l)) \end{pmatrix} \tag{3}$$

$$R_z(\psi) = \begin{pmatrix} \cos(\Delta\psi(l)) & -\sin(\Delta\psi(l)) & 0 \\ \sin(\Delta\psi(l)) & \cos(\Delta\psi(l)) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

The global displacement vector is calculated from discrete incremental displacement estimations, $\Delta x(k), \Delta y(k), \Delta z(k)$ which are obtained from two successive integrations of the accelerometer signals, at an arbitrary time integration interval Δt, as well as the estimated vector of discrete incremental spatial orientation, given by $\Delta\phi(k), \Delta\theta(k), \Delta\psi(k)$ [11].

B. *Numerical Method for Computing Velocity and Position*

In order to obtain the position and velocity, the numerical integration method was used to estimate the acceleration signals as function of time. These parameters of the particle are, respectively, x(t), v(t) and a(t). In digital signal processing, the acceleration data provided by sensor is not processed as an analytic function, but as discrete series of numerical values (Mello et al., 2015).

The Fig. 1 demonstrate the area under the curve between any two points can be approximated by as trapezoid, with area equal to Eq. (5)

$$A = \frac{(a_i + a_{i+1})\Delta t}{2} \tag{5}$$

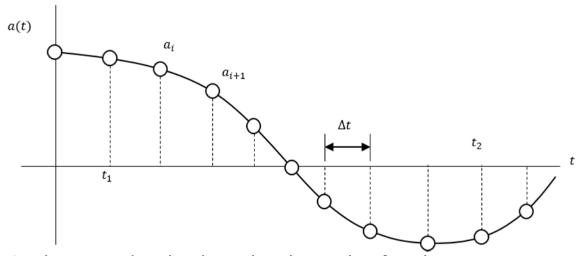

Fig. 1. Linear acceleration in each point as sine function.

The addition of such areas from some arbitrary times t_1 and t_2 yields the following estimate showed in Eq. 6:

$$v(t_2) - v(t_1) = \int_{t1}^{t2} a(t)dt = \sum_{t1}^{t2} \frac{1}{2}(a_i + a_{i+1})\Delta t \qquad (6)$$

In practice, the integration of the accelerometer signal results in a velocity signal that drifts over time due to the DC component in sensor measurement, which increases the position estimation error. As solution, a modulation technique is applied. The technique is based in the application of a voltage signal in the robot actuator with a sinusoidal wave shape where its amplitude is modulated by the desired control signal, causing the linear velocity of the robot behave as a sine function. Thus, the acceleration measured by the sensor also has a sinusoidal shape as shown in Fig. 1, and therefore, making it easier to identify the intervals of movement, where each step of the robot corresponds to a cycle of a sine function.

Each cycle of movement is integrated, rejecting any signal that behaves differently than the expected sine function(e.g, when the robot is not moving), When one integrates the acceleration, the angle of the velocity drift, due to DC value of the acceleration signal, is corrected to be integrated to estimate displacement.

A Low-Pass Butterworth Filter processes the acceleration measurements for each axis. Then, an acceleration threshold is applied in the measurements. If a sample is below of the threshold, then the velocity related to the sample is set to zero and it is inserted to a stack, however, if it is above the threshold, the step identification is executed. After that, the acceleration measurements between the identified intervals are integrated in order to estimate the velocity signal performed by the robot. Nevertheless, the velocity estimated is drifted due to the DC bias of the sensor, and this needs to be corrected. Thus, the corrected velocity signal is inserted to a stack. This procedure is repeated for each identified step. Lastly, the velocity array is integrated in order to obtain the displacement of the robot for each axis.

The Fig. 2 shows the filtered acceleration signal, the velocity signal and its bias correction, as executed by the algorithm to estimate the displacement.

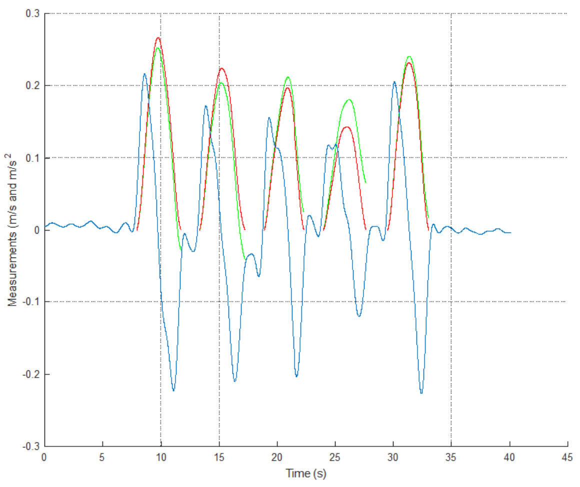

Fig. 2. acceleration, velocity signal and bias correction of the sensor.

C. *Orientation and Trajectory Generation*

In subsection *II - B*, while the accelerometer data processing is responsible to estimate the displacement, the gyroscope stands for the orientation estimation of the robot. The straightforward method to compute the orientation in each axis is by integrating the angular velocity measured by the gyro, as showed in Equation 7:

$$\theta_g = \int_{t1}^{t2} \omega_g \cdot dt \qquad (7)$$

with the displacement and the orientation estimated, the trajectory estimation can be calculated using the Eq. 8:

$$\begin{bmatrix} X(k) \\ Y(k) \\ Z(k) \end{bmatrix} = \begin{bmatrix} X(k-1) \\ Y(k-1) \\ Z(k-1) \end{bmatrix} + \prod_{k=1}^{n} R(k) \begin{bmatrix} \Delta x(k) \\ \Delta y(k) \\ \Delta z(k) \end{bmatrix} \qquad (8)$$

where $\Delta n(k)$ is the displacement for each axis, and $R(k)$ is the rotation matrix about the Z axis. The output data of the sensor such as position *(x, y)*, velocity *(x, y)*, and orientation were used as parameters to neural network.

D. *Neural Network Design*

The first step in a neural network application is to define the network structure and number of neurons. To determine which structure is the most appropriate, the type or kind of the problem has to be identified. In this paper, the application of regularizing a given trajectory is characterized as a fitting problem.

In [7], a neural network is used to improve the estimation of a visual odometry application. The results suggest that a two-layer feed-forward network with sigmoid hidden neurons and linear output neurons is appropriate as shown is Fig. 3.

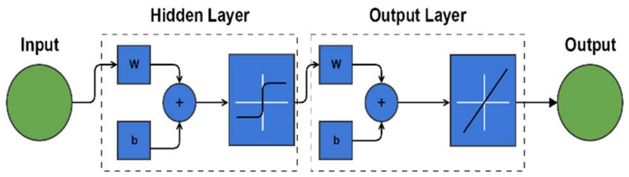

Fig. 3. Neural Network structure. Note that the first function applied is a sigmoid and then a linear function. The 'w' and 'b' represent the weights and biases of the network.

The Back Propagation algorithm uses the gradient descent rule to a feed-forward network works by optimizing the parameters of the network.

The optimal number of neurons should be determined experimentally, usually starting from 10. This number is not always directly proportional to the performance of the network. Sometimes increasing the number of neurons can increase the output error [1].

E. Neural network training

The neural network training is the step where the initial performance is determined. In order to setup this procedure, it is necessary an input vector which comes from the IMU and a reference vector, built with Kinovea positions as follows:

$$INPUT = \{x, y, psi, vx, vy\} \tag{9}$$

$$TARGET = \{xr, yr, psi, vx, vy\} \tag{10}$$

where x,y are the estimated positions (Inertial Odometry), xr,yr are the reference positions (Kinovea), psi is the orientation measured on the IMU and vx,vy are the linear velocities integrated from the acceleration provided by the IMU. Note that some parameters of the input and target vectors are equal, this mean that they are not regularized by the network but used for association. This process is indicated in Fig. 4.

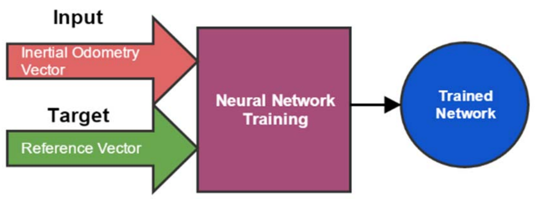

Fig. 4. Neural network training procedure

The input comes directly from the inertial Odometry estimation from the IMU on a mobile device. To obtain the target vector, the video corresponding to the same trajectory as the input vector has to be processed in Kinovea, as shown in Fig. 5.

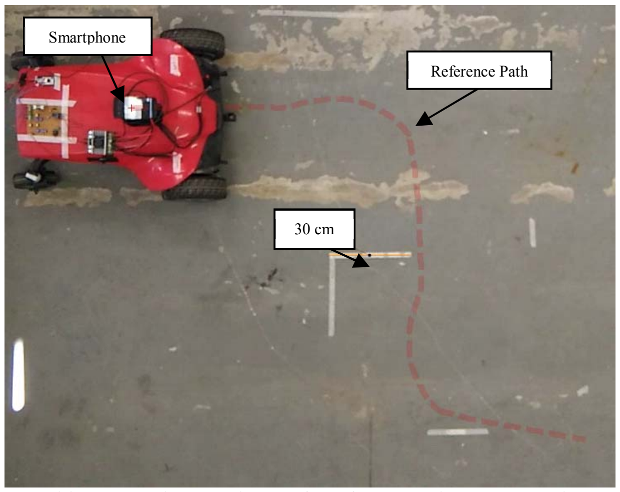

Fig. 5. Video processing on Kinovea for reference trajectory extraction. Note that there is a mark with known size in order to get the trajectory on the metric system.

There is an additional step before start training the neural network. Since the Kinovea calculates the trajectory on the image reference, the starting point is different, according to the robot's initial position. Setting the reference at (0,0) improves the network performance by avoiding the need of adaptation to different start positions.

After the input and target vectors are on the same reference, the another parameter needed to start training is the learning algorithm. This will determine how the network will adjust its weights and biases to fit the target vector. Common training algorithms are: Levenberg-Marquardt [10], Bayesian Regularization [8] and Scaled Conjugate Gradient [12]. Based on Lima, 2015 and Xu, 2009, in this application the training is performed with Bayesian Regularization.

Finally, the percentages of sampling for training, validation and testing are defined. The first quantity refers to the data used in the network adjustment (learning), followed by the portion used to measure network generalization (fit along all trajectory) and lastly the data used to measure the network performance during and after the training. Note that ideally, those data are divided using random indices. This also means that each training will have different performances due to the random sampling. Table I shows all the parameters used in the experiment.

TABLE I. NEURAL NETWORK TRAINING PARAMETERS

Parameter	Value
Input	Inertial Odometry Vector [position, orientation(IMU), linear velocities(integrated from IMU)]
Target	Reference Vector [trajectory (Kinovea), orientation(IMU), linear velocities(integrated from IMU)]
Number of Neurons	50
Learning Algorithm	Bayesian Regularization
Data Percentages	Training (70%), Validation (15%) and Testing (15%)

The application of the neural network is as follows:

$$H = F_{(1)}\left(INPUT \cdot \omega_1 + b_1\right) \tag{11}$$

$$O = F_{(2)}\left(H \cdot \omega_2 + b_2\right) \tag{12}$$

where ω_1 and ω_2 are weight matrices from input nodes to the hidden layer and from hidden layer to the output layer. $F_{(1)}$ and $F_{(2)}$ are the transfer functions of hidden layer and output layer respectively. H represents the hidden nodes and O the neural network output. The objective of the training process is to minimize the network error function defined by:

$$E = \sum_n \left|O - R\right|^2 \tag{13}$$

where R is the reference vector with the robot's current position.

F. *Addiotional Filtering*

The neural network regularized trajectory is noisy in some spots due to the inertial sensors precision and other calculation factors. To smooth the estimation it is applied an algorithm that watch the current variation in linear speed of the robot and the displacement in (x,y), as shown in (14).

```
if ((a<dvx<b && a<dvy<b) &&                 (14.1)
    (diff(1)>c || diff(2)>d))

    if (abs(diff(1)-diff(2)) > e)
                                            (14.2)
    ind=find(diff==max(diff))
        diff(ind)==maxdiff                  (14.3)
                                            (14.4)
            else
        diff(1,2)==maxdiff                  (14.5)

          end if                            (14.6)
        end if                              (14.7)
```

the variables dvx, dvy are the current differences in linear velocity; diff is the current displacement of positions (x,y); a, b, c, d, e, maxdiff are limits defined experimentally.

III. Results

The test was performed in a closed room with a workspace of approximately 2.35 meters by 3.78 meters. A camera recording at 1080p is setup on the top of the room in parallel to the floor. The robot is a Jazzy Elite® without the seat, shown in Fig 5. A smartphone is used as a sensor unit, recording data from its IMU during the trajectory. The video recorded is used to estimate the reference path using Kinovea's tracking, as illustrated also in Fig. 5. The velocities estimations integrated from the acceleration measured on the smartphone of test 1 is shown in Fig. 6.

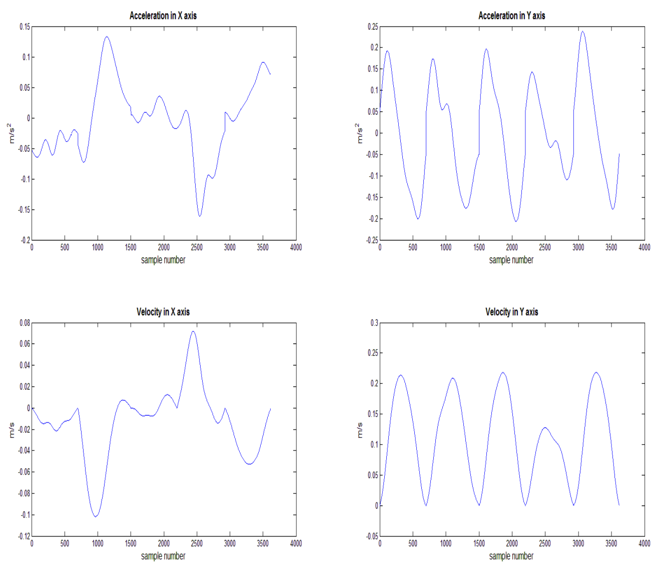

Fig. 6. Accelerations and velocities integrated from the smartphone accelerometer of test 1. Y axis is m/s or m/s^2 and X axis is the sample number.

Four tests were made with a single trajectory but from different starting points. The first test is used to train the neural network and the rest are used to validate the neural network performance, as shown in Fig 7. The reason for using only one test data for the training is to present the results of the neural network with minimal conditions.

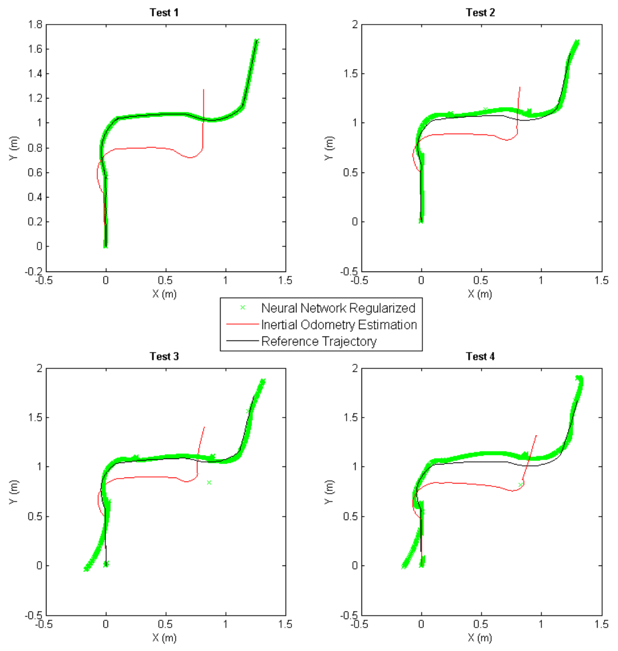

Fig. 7. Test trajectories. The first is the training and the remaining are the validation tests.

Visibly, the neural network improved the inertial odometry estimation. However, on tests 3 and 4 there is a drift from the starting point. The additional filter, described in algorithm (15) is applied to remove this effect, illustrated in Fig. 8

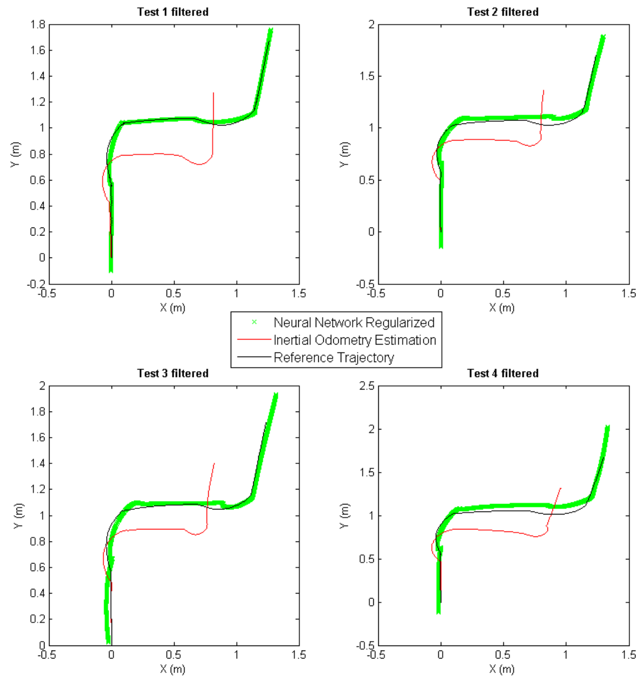

Fig. 8. Test trajectories with additional filtering.

The error is calculated based on the mean absolute error. Results comparing the inertial odometry only, neural network regularized and after the additional filter are shown in Table II.

TABLE II. Mean Absolute Error per Method

Test	Inertial Odometry		Neural Network Regularization		Neural Network plus Filtering	
	x	*y*	*x*	*y*	*x*	*y*
1	0.44	0.42	2e-4	8e-4	0.01	0.09
2	0.41	0.33	0.06	0.12	0.06	0.18
3	0.41	0.31	0.08	0.16	0.08	0.22
4	0.34	0.35	1e-4	0.22	0.03	0.35

IV. Conclusions

This paper presented the performance of a neural network application in improving the inertial odometry estimation for further position control implementation, mapping and localization of a robot with a single sensor unit (IMU). Even though the results are satisfactory for a single trajectory, the performance can possibly improve with more training data, a better post filtering algorithm and other training algorithms. The next step of this research is to test the neural network with more types of trajectories and longer paths, also the optimization of the additional post filter, use more sensors in order to verify if the total standard deviation of the measurements is decreased and application in 3D displacementAuthors and Affiliations

References

[1] Bohte, Sander M., Joost N. Kok, and Han La Poutre. "Error-backpropagation in temporally encoded networks of spiking neurons." *Neurocomputing* 48.1 (2002): 17-37. *(references)*

[2] Chen, Min, Shiwen Mao, and Yunhao Liu. "Big data: a survey." *Mobile Networks and Applications* 19.2 (2014): 171-209.

[3] Choi, Won-Seok, and Se-Young Oh. "Range sensor-based robot localization using neural network." *Control, Automation and Systems, 2007. ICCAS'07. International Conference on*. IEEE, 2007.

[4] Fourati, Hassen. "Heterogeneous data fusion algorithm for pedestrian navigation via foot-mounted inertial measurement unit and complementary filter." IEEE Transactions on Instrumentation and Measurement 64.1 (2015): 221-229.

[5] Guizzo, Erico. "How google's self-driving car works." *IEEE Spectrum Online, October* 18 (2011)..

[6] Jan, Racko, Brida Peter, and Machaj Juraj. "Comparison of output data from inertial sensors in smartphones." ELEKTRO, 2016. IEEE, 2016.

[7] Lima, Marcus VP, et al. "vSlam experiments in a custom simulated environment." *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*. IEEE, 2015.

[8] MacKay, David JC. "Bayesian interpolation." *Neural computation* 4.3 (1992): 415-447 *(references)*

[9] Marani, Giacomo, Song K. Choi, and Junku Yuh. "Underwater autonomous manipulation for intervention missions AUVs." *Ocean Engineering* 36.1 (2009): 15-23.

[10] Marquardt, D., "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *SIAM Journal on Applied Mathematics*, Vol. 11, No. 2, June 1963, pp. 431–441.

[11] Mello, Pedro Ramon, César Henrique Córdova Quiroz, and Paulo Roberto Gardel Kurka. "Navigation of a Mobile Robot Using a Device Based on Android Platform as an Inertial Measurement Unit" *XII DINAME,* 2015.

[12] Møller, Martin Fodslette. "A scaled conjugate gradient algorithm for fast supervised learning." *Neural networks* 6.4 (1993): 525-533.

[13] Oskiper, Taragay, et al. "Visual odometry system using multiple stereo cameras and inertial measurement unit." 2007 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2007.

[14] Pedley, Mark. "Tilt sensing using a three-axis accelerometer." *Freescale Semiconductor Application Note* (2013): 2012-2013.

[15] Siciliano, Bruno, and Oussama Khatib, eds. *Springer handbook of robotics*. Springer Science & Business Media, 484-490 (2008).

[16] Tomic, Teodor, et al. "Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue." *IEEE robotics & automation magazine* 19.3 (2012): 46-56.

[17] Xu, Haoming, and John James Collins. "Estimating the odometry error of a mobile robot by neural networks." *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*. IEEE, 2009.

[18] Zhang, G. X. "A study on the Abbe principle and Abbe error." *CIRP Annals-Manufacturing Technology* 38.1 (1989): 525-528.

[19] El-Raheem, R. M. A., Kamel, R. M., and Ali, M. F. (2015). Reliability of using Kinovea program in measuring dominant wrist joint range of motion. Trends in Applied Sciences Research, 10(4), 224.

[20] Kang, J. G., An, S. Y., and Oh, S. Y. (2010, July). Modified neural network aided EKF based SLAM for improving an accuracy of the feature map. In Neural Networks (IJCNN), The 2010 International Joint Conference on (pp. 1-7). IEEE.