Neural Network Based Movements for a Social Robot

João Sousa
Instituto de Sistemas e Robótica (ISR) Instituto Superior Técnico Lisbon, Portugal
joaopedromoreirasousa@tecnico.ulisboa.pt

***Abstract*—Mobile robots operating in the real world have the need to safely navigate through their surroundings. In order to achieve autonomy, it is imperative that these systems have good environmental awareness capabilities. In order to achieve this goal, the report presents the state-of-the-art literature on neural network applications in SLAM (Simultaneous Localization and Mapping). As an alternative to the already existing neural network SLAM implementations, an innovative approach to environment awareness and pose estimation is proposed, relying on LSTM (Long Short-Term Memory) recurrent neural networks organized in a Sequence-to-Sequence architecture. The used approach is divided into two parts as to simplify the overall problem: one LSTM network to predict the orientation, and another, inspired by neural machine translation methods, consisting in using a Sequence-to-Sequence network to directly map laser readings and motor velocity information into the robot's current coordinates. The experimental results obtained show that this architecture presents the generalization capability of mapping robot environment awareness and motor signals into its pose, extending its known ability of translation from one idiom to another. Finally, we devise several filtering steps that allow us to refine the networks' predictions, leading to more accurate results.**

***Index Terms*—Mobile Robots, Recurrent Neural Networks, Sequence-to-Sequence, Encoder-Decoder**

I. Introduction

A. Motivation and Problem Definition

In today's world, the constant growth of robot applications can be easily seen, whether it be for industrial, mobility or social purposes. One of the types of robots with a particular interest in being further developed is autonomous robots. However, autonomous navigation poses some challenges such as the possibility of frequent changes in the environment.

These issues are aggravated when taking into consideration a social robot situation, for which having poor obstacle avoidance and surroundings recognition performance would damage the desired natural interaction [1]. To ensure the most natural interaction possible, it is also desired that the robot is able to perform the environment recognition step by itself. This operation, in most cases, needs to be performed accompanied by a person and slowly, as to ensure a correct construction of the map.

The main objective of this thesis is the development of an environment recognition system for the social robot MOnarCH, capable of predicting the robot's pose (orientation and position) in its environment based on sensory data. A crucial aspect is that this system presents an innovation, being implemented using LSTM (Long Short Term Memory) units, transposing their capabilities in another area of interest into the problem at hand. More details regarding this are discussed further along the document.

B. MOnarCH Robot

The MOnarCH robots are social robots designed to interact with children, staff and visitors, engaging them in educational and entertaining activities in the pediatric infirmary at the Instituto Português de Oncologia de Lisboa. These are omnidirectional robots, whose maneuverability is based on four Mecanum wheels [2]. One of the MOnarCH robots is depicted in Figure 1.

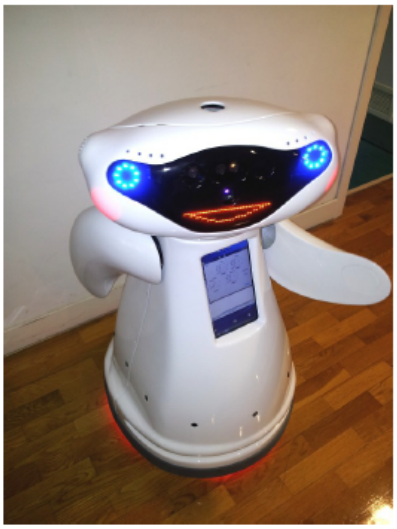

Fig. 1. MOnarCH SO robot [2].

In order to gain awareness of its surroundings and be able to navigate through them, the robot needs to collect data on the environment. By fusing the measurements provided by different sensors, the robot is able to perform such navigation. It disposes of a wide range of sensors, such as URG-04LX laser sensor to perform scanning and acquires the measurements from 683 points at a rate of 100 msec per scan.

MOnarCH robots possess two of these sensors, one in their front and one in their rear, enabling complete coverage of their immediate surroundings. The main functions of these lasers are mapping, localization, and object avoidance.

II. State of the Art

A. Neural Network Applications in SLAM

As previously stated, the task of constructing an efficient navigation system can be divided into two separate stages: environment awareness and task planning.

A good representation of the surroundings allows for an optimization of navigation instead of performing reactive actions based only on the most recent sensory data, since the whole map is unknown. To tackle this problem, SLAM (Simultaneous Localization and Mapping) methods were developed to obtain map representations and the robot's localization relative to it [3]. SLAM is a method in which robots equipped with sensors such as laser, vision and odometer, construct a map while understanding the unknown environment. Succinctly, the various sensors estimate the motion information of the robot and feature information of the unknown environment and by fusing the information an accurate estimation of the pose of the robot and spacial model of the scene is acquired [4] .

NN (Neural Networks) were inspired by the human brain and its capability of performing complex tasks in real-time. Just like the human brain, by connecting many of these neuron units, incredible performance is achieved, hence why the application of NN in a wide area of situations is a trend. The majority of NN applications in SLAM is related to the loop-closing problem. Loop-closing is the act of correctly asserting that a vehicle has returned to an already previously visited location. This task is hindered by the error accumulation prone to some sensors, such as odometry estimators, whose data is frequently used in map building, leading to lower map quality. Deep neural network can also be used to exploit cues generated from 3D LiDAR (Light Detection and Ranging) data [5]. Since as the robot navigates, map lines that should be parallel start getting skewed due to the error accumulation, this method estimates an image overlap generalized to range images and provides a relative yaw angle estimate between pair scans.

NN can also be applied in fusion with SLAM in order to classify encountered obstacles, such as creating a Semantic SLAM through the development of a convolutional neural network [6]. Other examples are performing semantic application of NNs in SLAM [7]. This method however classifies areas of the SLAM-generated map into the categories of: corridor, room or doorway.

Li et al. [8] proposed a Jacobian free NN based Fast SLAM that deals with odometry systematic errors and the SLAM's algorithm linearization errors, acting as a compensator.

From the above examples, it can be seen that a variety NN implementations in SLAM algorithm were done. However, they perform the task of error handling and correction in already constructed maps or feature addition to already constructed maps. By studying the applications of NN in other areas, one of those applications can be transposable to a direct environment recognition by the NNs.

B. Recurrent Neural Networks

In a globalized world like ours, the need to understand and access information in multiple languages is dire. Whether it be simply to allow people to communicate easier or to allow a person from another country to access information or knowledge from another, the computation of a language translation system can is very helpful to us. NMT (Neural Machine Translation) is the approach used nowadays to tackle this problem. It consists of using artificial neural networks that are suited for tasks that would require memory logic.

The most common model used in NMT is Sequence to Sequence Learning, which consists of an encoder-decoder architecture, both of which are implemented through LSTM (Long Short-Term Memory) [9].

C. Sequence To Sequence Model

As previously stated, Sequence-to-Sequence models, that fall under the machine learning model category, have proven to hold performance-wise on challenging problems such as visual object and speech recognition and also in NMT.

From a general point of view, a Seq2Seq (Sequence-to-Sequence) model is comprised by two main components: an encoder and a decoder. The encoder is responsible for taking in the input sequence and converting it into a state. In turn, the decoder is in charge of reading the state and producing the output. A representation of a Sequence to Sequence model is depicted in figure 2.

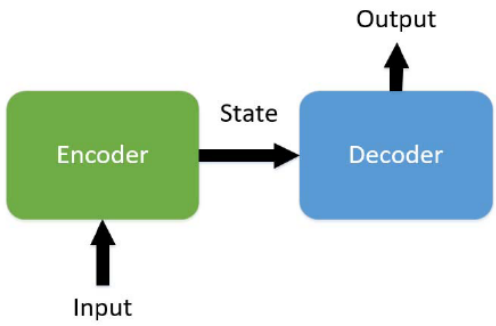

Fig. 2. Sequence to sequence model representation [10].

In order to better comprehend how these two components work, a deeper understanding on how RNN (Recurrent Neural Networks) work is needed.

1) Recurrent Neural Networks: Recurrent Neural Networks RNN is a type of network best suited to deal with sequential data. They are distinguished from other network architectures by their 'memory' mechanism. In contrast to other networks that assume the independence between inputs and outputs, RNNs take the assumption that current inputs and outputs depend on prior inputs from the network [11]. This 'memory' mechanism is then constructed by feeding the output of the unit cell to itself.

The effect of memory persistence is then achieved by computing a hidden state $\boldsymbol{h(t)}$ at each time step that takes

into account the previous state. That hidden state is updated according to:

$$h(t) = f(h(t-1), x) \tag{1}$$

where $\boldsymbol{f}$ is a non-linear activation function and $\boldsymbol{x}$ is a sequence with variable length. In terms of activation functions, these can vary from sigmoid function or hyperbolic tangent function to more complex such as LSTM units [10], [12].

LSTM networks are a specific type of RNN, which are widely used in the NMT problem due to their achieved results. The composition of a common LSTM unit is as follows: a cell, an input gate, a forget gate and an output gate. Due to these elements, these cells are known to be able to retain previous values for either long or short periods of time.

The main idea of LSTMs is that its cell state is regulated, that is, information can be added or removed. These gates are a way to optionally let information through and they are composed of a sigmoid neural network layer and a pointwise multiplication operation. As a result of the sigmoid operation, numbers between zero and 1 are outputted, describing how much of that component gets to go through. The closer to 1 the value is, the more relevant that component is in this time step.

Getting into more detail, the first step done in the LSTM unit is to select what information is thrown away from the cell state. That operation is made by the forget gate layer. This layer takes the values of $\boldsymbol{h_{t-1}}$ and $\boldsymbol{x_t}$, concatenates them, and passes them through a sigmoid layer that outputs values between 0 and 1 corresponding to each element present in the previous cell state $\boldsymbol{C_{t-1}}$. As mentioned before, these values represent the weight given to the previous cell state values, according to their importance to keep in future time steps [13].

The computations processed by this gate can be summed as:

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{2}$$

where $\boldsymbol{W_f}$ and $\boldsymbol{b_f}$ are the weights and the bias of the network composing the forget gate respectively.

The next phase allows it to select which new information will be stored in the cell state. This phase is comprised by two parts: firstly, a sigmoid layer, denominated input gate layer decides on which values to update; secondly, a tanh layer creates a vector with new candidate values, $\tilde{C}_t$, that could possibly be added to the state.

The operation computed on this step can be written as:

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{3}$$

$$\tilde{C}_t = tanh(W_C.[h_{t-1}, x_t] + b_C) \tag{4}$$

Following these steps, the update of the cell state can be computed. This operation is achieved by doing:

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{5}$$

Lastly, the output remains to be computed. The output consists of applying a filter to operations based on the current cell state. Firstly, a sigmoid layer is applied to the concatenation of the previous hidden state and the input. This result will determine which values of the cell state are going to be output. Then, the cell state is filtered by a tanh function, that results in the cell state's values comprising between -1 and 1. After, the output of the sigmoid layer is multiplied by the result of the tanh function. All of the steps that were mentioned before are demonstrated in Figure 3.

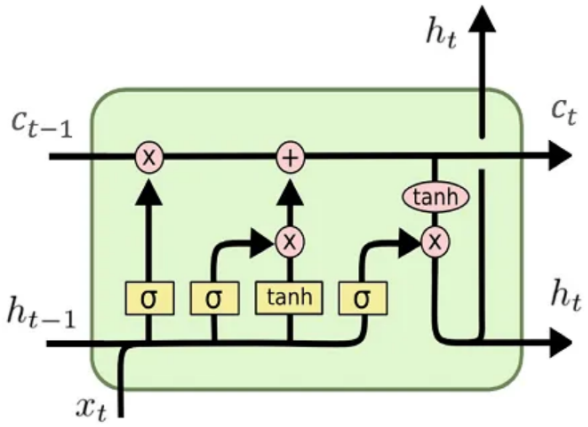

Fig. 3. Operations representation of an LSTM cell [10].

This last step can be represented by the following equations:

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{6}$$

$$h_t = o_t \times tanh(C_t) \tag{7}$$

2) Neural Machine Translation: Having seen the core element of NMT that is LSTM used in both encoder and decoders in depth, the application of a network of these units can now be studied.

In the NMT problem, a source sentence is fed to the network, in order for it to be translated into the objective language. Having the example used in [10] to translate English sentences into French, the encoder receives the source sentence followed by a marker for example "-" or "$< EOS >$" meant to signal the end of the sentence [12], [14]. Afterwards, the decoder generates the corresponding word translation, and this translation is then fed to the next decoder in order to improve the next word's translation, capturing a sense of context.

Usually, networks used in NMT have an embedding layer, as the first layer, whose goal is to receive the source sentence and generate the corresponding word representation that gets passed to the next hidden layers of the network. To finalize, the last layer of the decoder side is composed by a softmax layer. This enables the network to convert the results into probabilities for the words present in the vocabulary. An example of a simple architecture depicting this situation is represented in Figure 4.

As an improvement to the architecture shown, certain methods were developed to further increase the model's accuracy. The attention mechanism is one of those methods, allowing the

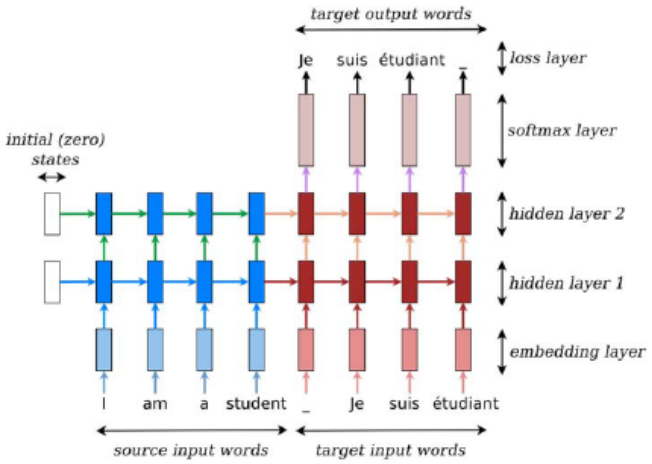

Fig. 4. Neural Machine Translation architecture using LSTMs [15].

enhancement of accuracy when sentences possessed increased length [16]. This mechanism can be defined into two categories according to where the attention is placed. Global attention assigns attention weights to all previous words while Local attention assigns scores only to words within a window with a fixed size. An underlying increase in memory usage occurs since we need to store previous context vectors, so a trade-off between accuracy and memory consumption must be made [12].

Another strategy is the bidirectional mechanism. This consists of the implementation of a backward and forward cells that scan the input sequence in a backward and forward direction. The idea behind this is that the output at one specific time does not only depend only on past elements but also on future words and by applying this, the model gets information on the left-side and right-side contexts. In order to achieve this, this architecture contains two RNNs on top of each other, and the final output is computed taking into account the hidden states present in both those networks [17].

D. RNN results on Neural Machine Translation

Next, the performance of these models will be discussed, as well as the data set size and other parameters.

Firstly, Luong et al. [14] implemented an English-German translating system. The used dataset was composed of 4.5 million sentence pairs mostly from formal texts coming from the WMT tournament, to which were added 200000 sentence pairs, consisting of spoken language provided by IWSLT competition 2015. The chosen model implemented was Seq2Seq LSMT with attention.

Tiwari et al. [12] experimented with English-Hindi translation. In those experiments, the CFILT (Center for Indian Language Technology) IIT Bombay English-Hindi parallel corpus [18] was used, together with Digital Research Infrastructure for the Language Technologies, Arts and Humanities HindEnCorp. These datasets when merged were composed of 509442 sentence pairs that were split into 377884, 94470, and 118088 sentence pairs training, validation and testing data respectively. The implemented models were uni and bidirectional Seq2Seq LSTM.

Khan et al. [17] performed Urdu transliteration from Roman script to Urdu script. It is to be noted that, the characters of words on both sides of this translation are not the same. In fact, there is no common character in them. In terms of the data set, a 53800 sentence pair corpus was generated which was randomly split into training, validation and testing sets having the following size respectively, 45500, 3300, 5000. Three different architectures were tested.

Lastly, Bonilla et al. [10] implemented an automatic translation of Spanish commands to Control Robot Commands. This approach translates simple movement commands in Spanish to Robot Control Language which is a simple programming-like language consisting of atomic instructions pointed to the execution of a minimal and well-defined task. A small dataset was used composed of 3785 lines in both Spanish and RCL (Robot Control Language). For this application, three models were used: Simple Seq2Seq, with attention mechanism and a bidirectional network.

In Table I, a summary of the performance scores of the aforementioned applications of Seq2Seq architectures in NMT can be observed.

TABLE I
SUMMARY OF THE PERFORMANCES OF NMT APPLICATION STUDIED IN THIS CHAPTER.

Model	Language	BLEU	Loss
Seq2Seq LSTM with Attention [14]	English-German	31.4	-
Simple Seq2Seq LSTM, Bidirectional Seq2Seq LSTM [12]	English-Hindi	14.9923 15.2034	-
Simple Seq2Seq LSTM, Bidirectional Seq2Seq LSTM Bidirectional Seq2Seq LSTM with Attention [17]	Roman Urdu - Urdu script	64.99 83.38 83.39	-
Simple Seq2Seq LSTM, Bidirectional Seq2Seq LSTM Bidirectional Seq2Seq LSTM with Attention [17]	Urdu script - Roman Urdu	79.6 89.15 89.15	-
Simple Seq2Seq LSTM, Bidirectional Seq2Seq LSTM Bidirectional Seq2Seq LSTM with Attention [17]	Spanish - RCL	-	2.83e-0.8 0.08 3.18e-0.8

E. Discussion

Although since the specific objective of the models discussed was different and therefore the datasets used were also different, some conclusions can be reached. In general, these models show a high demand of data in order to be trained. Depending on the complexity of some of the objective languages of the translation, it has been shown that the data corpus size can vary: 4.7 million in [14], to about 500 thousand in [12] and 49 thousand in [17].

As the majority of the examples seen, these models are mostly used to perform translation between languages that use the same alphabet [14], [19]. There were also applications on transcriptions between languages with different alphabets [12], [17]. There have also been applications in translating between a spoken language (Spanish) and simplified programming commands (RCL) [10]. This shows that there is potential in Seq2Seq LSTM networks to perform translating, mapping to be more general, tasks than simply between languages.

The task of recognition of one's surrounding environment, whether it be an autonomous vehicle or a person, relies on

detecting the topology of the environment, while keeping track of the previous landmarks and the displacement done. This task has the need of a 'memory' component, that allows to take conclusions of the new localization based on previous ones. The innovation idea of this paper lies in the study of the application of Seq2Seq LSTM networks, to perform this environment awareness. The idea is to apply laser range signals and the robots' motor velocities as the network's input, enabling the network to perform a map of this data into the robot's 2D pose, composed of x position, y position and angular orientation.

III. Methodology

In this section, the methodology used in tackling this problem will be described, as well as all the information found pertinent to its solution.

Firstly, due to the complexity of the problem before us, it was decided that, instead of using a Seq2Seq Encoder-Decoder model to estimate both the x and y position and the robot's orientation, the problem could be divided into two subproblems: orientation estimation and position estimation.

A. Data Acquisition

The data acquired to perform the tests was obtained by building an environment within the laboratory and using the MOnarCh robot to navigate through it. The built environment was 2.9 meters wide for 5.3 meters long having a cube in the middle serving as an obstacle. In the environment's periphery, there were some other household objects such as a sofa and some furniture. This is illustrated in Figure 5.

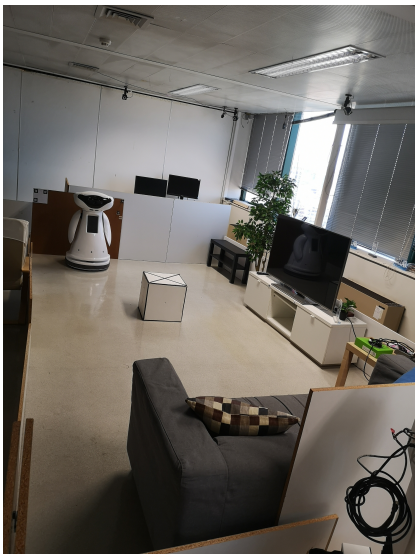

Fig. 5. Data acquisition environment

In order to take into account different kinds of stimuli coming from the robot's movement, this environment was navigated not only in straight lines but also recurring to varied curved movements.

B. Data Structure and Preprocessing

As previously referred, the data used for the model's training must be acquired through the robot's odometry and laser data. This data, however, has to suffer transformations in order to better address the problem.

Taking this into account, the MOnarCh robot's topics used were as follows:

- *Odometry* with an acquisition rate of 20 Hz, from which we obtain the odometry estimation x and y value data, the orientation estimation in Quaternions, the linear x and y velocity, the angular velocity regarding the z axis and the time stamps for every acquisition.
- *Scan* with an acquisition rate of 20 Hz, from which we obtain 689 laser range values with a minimum range of $0.023m$ and maximum range of $60m$. These lasers are located in the front part of the robot spread throughout $180°$. Each acquisition time is also saved.
- *Scan2* with an acquisition rate of 10 Hz, from which we obtain 513 laser range values with a minimum range of $0.019m$ and a maximum range of $5.59m$. These lasers are located in the rear part of the robot spread throughout $180°$. Each acquisition time is also saved.
- *Idmind_motors/encoders* with an acquisition rate of 20 Hz, from which we obtain the number of ticks each wheel has from each acquisition.

The robot's laser scan data, together with the odometry data are used as input into the networks, while the encoder's data is used as the ground truth for training.

1) Input Data: As aforementioned, the data serving as input into the NN must suffer some transformations. The first transformation comes from the fact of not all ROS-topics having the same data acquisition rate. So, in order to overcome this while taking into account that each row of the data says respect to the same moment in time, the data needs to be synchronized. This was accomplished by inputting both the data vector with the higher rate and the least, for each instance of acquired data both time stamps are compared, and only when this value surpasses for the first time the value of the one with the lower rate, is the information saved. This way, we end up with data vectors of all topics with the same size and relative to the same point in time. By applying this to the acquired data, all topics' data synchronize with the topic *Scan2* with an acquisition rate of 10 Hz.

Another important transformation that was implemented was transforming the robot's orientation information from Quaternions into degrees, since, due the sought application, degrees being easier to identify the robot's orientation in the environment. The yaw in degrees can then be obtained through the following expression:

$$\psi = \frac{180}{\pi} tan_2(2(q_w * q_z + q_x * q_y), 1 - 2(q_x + q_y)) \quad (8)$$

$$q_w = cos(\alpha/2) \quad (9)$$

$$q_x = sin(\alpha/2)cos(\beta_x) \tag{10}$$

$$q_y = sin(\alpha/2)cos(\beta_y) \tag{11}$$

$$q_z = sin(\alpha/2)cos(\beta_z) \tag{12}$$

where α is a simple rotation angle and $cos(\beta_x)$, $cos(\beta_y)$ and $cos(\beta_z)$ represent the direction cosines of the angles between the coordinate axes and the axis of rotation.

After performing these various transformations, the data is ready to be brought together. This aggregation is done through the concatenation of the various measures into a vector. The representation of one data instance is shown below:

$$input = [scan_{front}, scan_{back}, x, y, \theta, v_x, v_y, t_s] \tag{13}$$

To reduce the complexity of the input data, the data points dimension considered for $scan_{front}$ and $scan_{back}$ was reduced. To give more relevance to the laser data coming from the robot's front, only every 5 laser data points were used. The same method was applied to the laser data coming from the robot's rear, but to give it less relevance, only every 10 data points were included. these were concatenated with the odometry's x and y positions, the calculated orientation in degrees, linear velocity regarding the x and y axes, and lastly a measurement timestamp. This results in a vector that has 197 in length for each acquisition time.

2) Ground truth Data: In this problem's case, the ground truth data is derived from the robot's encoders. Encoders are known to accumulate error over time due to phenomena such as wheel drifting. For this reason, the use of encoders as ground truth data on these experiences leads to the results being applied for short-term navigation. Each encoder's measurement coming from the topic *Idmind_motors/encoders* gives us the number of ticks each of the 4 encoders go through each time step. According to [2], each encoder contains 27592 slits, meaning that each encoder has a resolution of 0.047°. Using the encoder information to calculate the angular velocity of each wheel, the evolution of the robot's position can be obtained through its forward kinematics expression:

$$\begin{pmatrix} v_x \\ v_y \\ \dot{\theta} \end{pmatrix} = \frac{r}{4} \begin{pmatrix} +1 & +1 & +1 & +1 \\ -1 & +1 & +1 & -1 \\ \frac{1}{L} & \frac{1}{L} & \frac{1}{L} & \frac{1}{L} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} \tag{14}$$

in which r represents each motor wheel radius, L is the sum of the x-axis and y-axis distances of the wheel center to the robot frame's origin and w_i represents the angular velocity of each wheel.

Taking this dynamic relationship into consideration, and knowing the time taken between each data acquisition, it is possible to calculate the robot's x, y and θ displacement within each time-frame and subsequently, by summing all these displacements, determine its overall pose coordinates regarding its starting point.

Our approach to the problem was divided into two smaller problems due to its complexity. It was decided that 2 different models would be built: one focusing on the determination of the robot's orientation θ throughout its motion and another model focusing on the determination of its x and y location coordinates of the world. A high-level diagram of the proposed solution is shown in Figure 6.

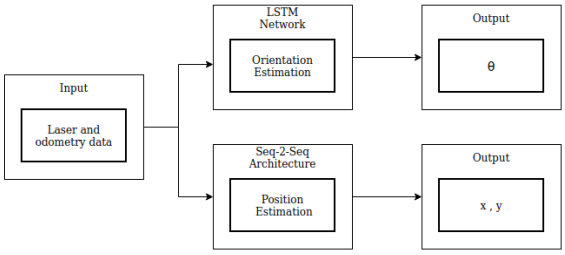

Fig. 6. High-level pipeline of the implementation

C. Orientation Estimation

In order to implement the orientation estimation, a NN comprised of LSTM cells was built. As the consequent orientation changes of the robot in space are dependent on its previous values, RNN was used due to its capacity on capturing the data's sequential meaning.

To give emphasis to the sequential nature of the data, the data is presented to the network through a moving window with a size of 7, also containing the last 6 computed time steps.

This network takes a 7 by 197 matrix as its input, for each time step and it is composed by various layers. The input layer represents the format of data being inputted in the 11 LSTM cells layer, followed by another layer of 5 LSTM cells and finally, a layer containing one cell. This is in turn followed by one dense layer that uses linear as the activation function in order to extract the result. The logic behind these sequential layers is that the NN can gather information and pass the acquired feature information onto the next layer, so that at the end the network can output an orientation estimation. The values chosen for the number of layers and number of LSTM cells per layer were chosen through trial and error and observing the results.

The network was trained using mean squared error as its loss function and 50 training epochs.

D. Position Estimation

As referred to previously, the second problem to be tackled is the robot's position estimation. For this problem, a generalization of NMT methods presented in section II is used, in an attempt to test the generalization capability of these networks to map a set of data containing laser ranges and odometry values.

Since we are transposing our problem into the NMT solution paradigm, further transformations must be applied to the data. These types solutions have words as a unit, which in turn are

used to build a dictionary composed of all the unique words found in the data set.

In this specific case, these words will be numbers and, as the robot's sensor data has a high range capacity and possess many decimal places to account for precision, some considerations are made. Firstly, the number of decimal places from the data need to be reduced, as values with the same digits until, for example the $1e^{-5}$ would be considered different words in the dictionary due to the lower decimal places having different digits. To overcome this we multiplied all values by 100, proceeding with rounding all values in regard to the unit and, lastly, clipping all values below -999 and above 999. These transformations allow for the reduction of the data set range, comprehending values between $-999\ cm$ until $999\ cm$. This way the data has accuracy to the centimeter level.

Following these transformations, the data was converted into the *string* format and two tokens had to be added to the target data for training: the 'START_' token and the '_END' token.

1) Encoder-Decoder Architecture with Teacher-forcing: As illustrated in Figure 4, Seq2Seq encoder-decoder architectures used in NMT, take as input in the decoder part a shifted copy of its output. For this reason, the decoder's behavior changes during training and testing phase.

The encoder takes the network's input and each cell produces the internal state passed into the decoder to initialize its internal states. The decoder in turn receives as input a shifted version of its output, starting with some initialization token.

This fact makes it difficult during training, as the decoder is being trained, having its own shifted output as input, meaning that the weight change performed during backpropagation is not being done on accurate ground truth data.

The teacher-forcing method consists of setting as input of the decoder shifted the ground truth training data, having an initialization token at the beginning instead of the predicted output. This method facilitates the training of the decoder cells.

Similarly to the orientation solution, the network's structure begins with its input layer. There are two input layers, one corresponding to the encoder and the other corresponding to the decoder. Following each of these layers, there are also embedding layers, which serve as an alternative to one-hot encoding, but instead of having 0's and 1's it has real values. The first embedding layer connects to the encoder comprised of 100 cells, which in turn has as output the two internal states and its output. In turn, the decoder receives as input the embedded ground truth data and the two inner states from the encoder. Lastly, the output from the decoder is passed onto a Dense layer, whose output is in the form of two vectors, one for the x and the other for the y coordinates. The values chosen for the number of cells for each layer were chosen through trial and error and observing the results.

After training these layers, the encoder layer remained the same, and the trained weights from the decoder were used to build a new decoder, where the input begins with the 'START_' token followed by the decoder's prediction for the previous time step.

IV. Results

In this section, the results derived from the experiences performed on the networks referred on Section III will be shown, together with a critical analysis of said results.

A. Orientation Estimation

Using the LSTM NN architecture, after performing training, and applying the same data preparation to the test data set, the network produced results, whose zoomed-in data can be seen in Figure 7.

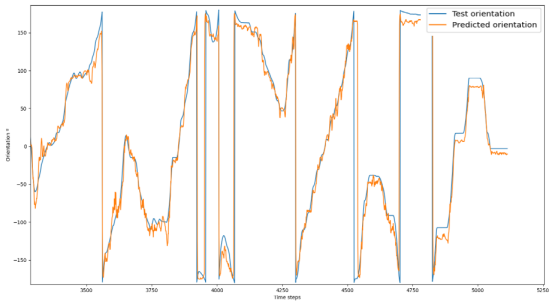

Fig. 7. Ground truth orientation (blue) vs predicted robot orientation (orange) - zoomed in

As observed, the predicted orientation is presented with a lot of jittering. In order to smooth the orientation prediction, the prediction can be passed through a median filter with a window of size 15. By applying that median filter, the orientation overall estimate becomes as pictured in Figure 8.

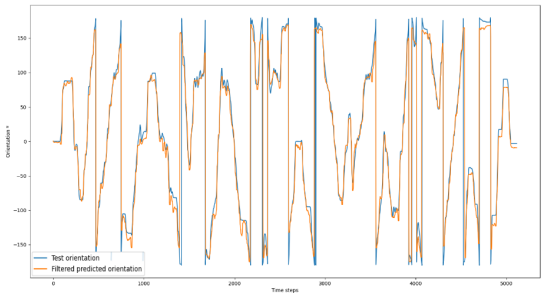

Fig. 8. Ground truth robot orientation (blue) vs filtered predicted robot orientation (orange)

1) Further Test: Since the used pipeline contains the robot's odometry as the network's input and the desired result is to get a corrected orientation estimation, a test was performed as to how this architecture behaves when exposed to more error. In order to do this, the same architecture was trained using the same conditions, being the only changed variable the fact that a linear error is added to the ground truth data of the model corresponding to 0.002 radians per time step.

The results of such test are provided in Figure 9. It can be seen that applying the linear error to the neural network's data hinders its performance and the prediction discrepancy is more notorious at the end due to the error accumulated from

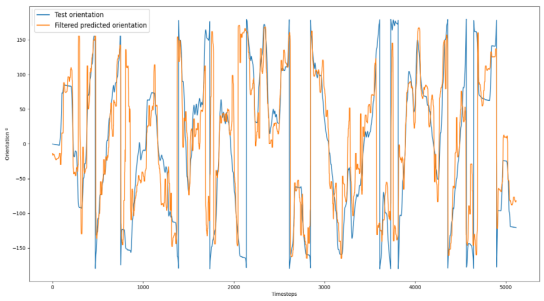

Fig. 9. Ground truth orientation (blue) vs filtered predicted robot orientation (orange) with added error

the previous time steps. Though it can also be affirmed that the network is somewhat capable of following the movement's dynamic.

B. Position Estimation

In this subsection, the results obtained from the position estimation solution are shown. Recurring to the weights trained in the architecture referred to before, the inference mode was used in the test data set.

As observed from the close-up of the results, the prediction presents itself with noise in the form of spikes throughout the whole inference period. as shown in Figure 10 and Figure 11. It can also be seen through the origin of the various noise spikes, that the prediction is able to follow the dynamic of the ground truth data.

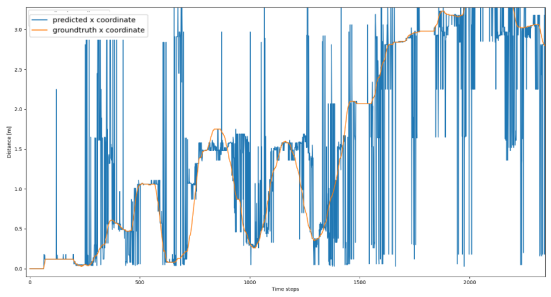

Fig. 10. x coordinate prediction (blue) vs ground truth (orange) - zoomed in

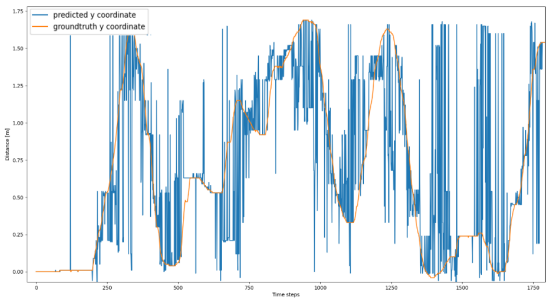

Fig. 11. y coordinate prediction (blue) vs ground truth (orange) - zoomed in

Taking this fact in mind, the predictions were passed through a median filter with a window of size 25 with the finality of attenuating the noise present, achieving what is present in Figures 12 and Figure 13.

Fig. 12. x coordinate prediction filtered (blue) vs ground truth (orange)

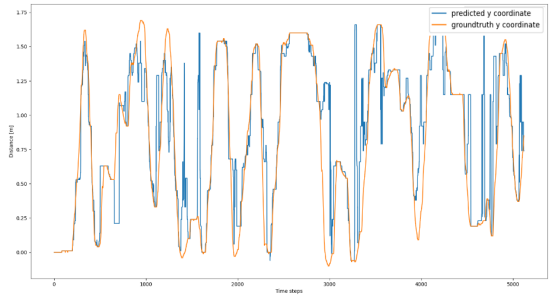

Fig. 13. y coordinate prediction filtered (blue) vs ground truth (orange)

After this filtering step, both coordinate predictions become clearer, allowing the predicted results to follow more faithfully the ground truth data. Nevertheless, there are still some spikes present in both predictions, showing intervals where the signal has a fast-changing derivative. By taking the values of the changing derivative and together with knowledge regarding the robot's physical model, a further filtering step can be implemented, attenuating the influence of these fast-changing derivative values.

At the time of the data sets acquisition, the MOnarCH robot's speed was set to 1 meters per second. Knowing that all acquisition rates were equalized at 10 Hz, one can infer that the maximum displacement the robot could have between subsequent time steps would be of 0.1 meters. With this knowledge, a filtering step was added, in which the discrete difference between points was calculated within one moving window, and if the difference between values was higher than a predefined slope value, it would be substituted by the mean value of that same window interval. In this case, the maximum threshold slope value was set to 0.1 to be in accordance with the information from the robot's physical model, and both x and y coordinates were filtered using a 65-sized window, achieving the predictions in Figure 14 and Figure 15.

To complement the comparison between prediction and real values, the root mean squared error of both coordinates was calculated for the entire testing interval, having x the root mean squared value of 0.31688 and y having achieved 0.28260, both rounded to the fifth decimal place.

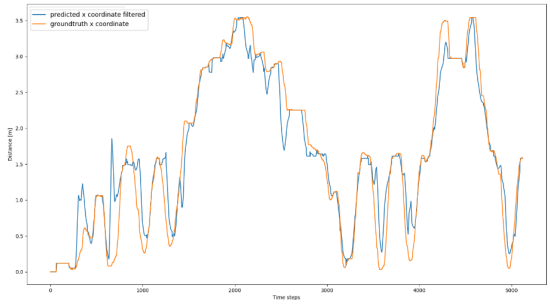

Fig. 14. x coordinate prediction spike filtered (blue) vs ground truth (orange)

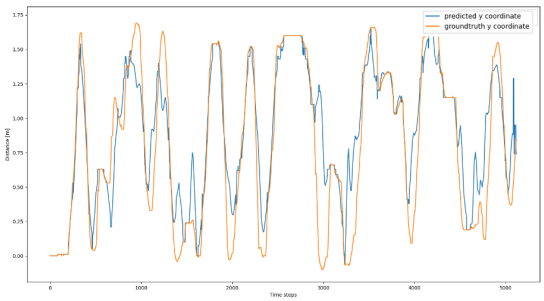

Fig. 15. y coordinate prediction spike filtered (blue) vs ground truth (orange)

C. Discussion of Results

In this section, we analyze the results obtained from the performed experiments.

Regarding the orientation estimation, by observing Figure 7 a view of the robot's predicted orientation and its true result can be seen throughout part of navigation duration. From this picture alone, it seems that the LSTM NN result is capable of accompanying the general dynamic of the ground truth. Also, at the points where there is a rapid change in value is where the network is not able to accompany the ground truth as accurately. The majority of these points represent a sudden change, as the $360°$ around the robot are set in an interval of $[-180°, 180°[$, making it so that when changing from one interval limit to the other value change is abrupt.

To tackle the fact that the result presented jittering , a filtering phase was added, by applying a median filter. Through trial and error, the chosen filter configuration was a filter of window size 15, achieving the results depicted in Figure 8. By inspection of these two Figures, it can be said that the predicted signal achieved a higher smoothness level. The trade-off of using this filtering step is that now, it is more evident that the lack of capacity from the model on following abrupt signal changes originated from passages of one orientation interval limit to another.

Although the odometry data contained in the input data already possess errors coming from drifting from the robot's wheels, a further test was performed to comprehend the capacity of these networks to accommodate error handling. To that end, the same architecture was used, but trained on altered ground truth data, with an added linear error of 0.002 radians per time step. Results from this experiment are shown in Figure 9, which presents a decrease in the network's capability of predicting the orientation.

Regarding the position coordinates estimation, the use of the Seq2Seq architecture by itself is not able to achieve reasonable results, as shown in Figures 10 and 11. Once again, by close inspection of both results, we are able to understand that the predictions follow the desired dynamic, although with a lot of noise throughout the prediction interval. With this in mind, the same strategy of implementing a median filter was done. Both results were passed through a median filter with a 25-time step size, reaching the results in Figures 12 and 13. With these results, the capability of predicting the robot's position shows improved results, still possessing some points where the prediction differs with a high derivative in a short interval. To that end, by adopting an additional filtering step combined with knowledge of the robot's physical model, a filter that attenuates slopes that exceed a certain threshold according to the robot's kinematics is implemented. Those results are shown in Figures 14 and 15. The obtained results show an increased improvement when compared to the performance shown by using the Encoder-Decoder on its own, having both coordinates a low root mean squared error value.

On both our sub-problem solutions, the acquired data is presented with a lot and variable stimuli, resulting in signals that have aggressive changes and possess difficult patterns to follow, leading to increased difficulty in model training. Regarding the innovation of the use of NMT techniques in the robot map-making domain, we are challenging the generalization capability of this architecture even further than it is originally made. In the NMT domain, all sentences are translated from one language to another, but there hasn't got to be an existing relationship between sequential sentences, whereas, in the innovative use domain, all mapping instances are intimately related to the previous. This is due to constraints of physical nature, as robot's successive positions have a continuity relationship.

D. Model constraints

During the training process of the position estimation problem, further data transformations were made, in an attempt to reduce the scope of values originated by the sensors. This reduction was also in aid to the reduction of the dimension of dictionaries built, that are used in the model training. The direct consequence of limiting the incoming data to the interval of $[-9,99; 9,99]$ meters is the limitation of the area that the robot can navigate using this system.

Regarding the time spent on training both these models, they both took 5992 time steps of data for training. The orientation solution ran its training phase for about 1 hour, while the position coordinates Seq2Seq network took roughly 2.5 hours to train its weights.

Furthermore, in the training and testing phases, the robot started in the same position for both data sets collection. This translates into that our conclusions here can only be inferred

for those cases, where the robot starts in the same station locations on startup. For other cases, the navigation strategy as-is should prove inefficiency.

Additionally, by using extra filtering steps to smooth and enhance results obtained from the networks, we are adding processing steps that increase the time needed to acquire the final result of the pose estimation.

Another strong argument is the use of encoder data as ground truth for the model. For long navigation times, encoder data is known to get skewed by phenomena such as wheel drift. For this reason, encoder data by itself is only accurate for short periods of time, since it hasn't accumulated much error. Since the objective of this work was to evaluate the capability of Seq2Seq networks to map environment sensory data into position, the use of encoder data as ground truth allows us to perform such analysis. However, for this solution to be usable in cases of prolonged navigation periods, it should be fused with positioning error correction strategies, such as the use of sparse landmarks in the environment. By using these two strategies together, the robot would be able to navigate using the output of the implemented models, and when passing by one of these landmarks, whose exact location is known, the robot's position would be updated, correcting the possible errors acquired through its navigation period.

V. Conclusions

In this work, we proposed an LSTM-based solution to be applied to the robot's SLAM problem. The proposed pipeline is divided to tackle two sub-problems separately: orientation estimation and position estimation. On the former, an LSTM neural network was built, relying on the memory capacity of these cells to capture sequential meaning of data. Regarding the latter, a Seq2Seq Encoder-Decoder architecture was built. This solution presents scientific novelty, as the proposed solution is inspired in an architecture related to NMT problems, not having been used in the paradigm of robot's pose estimation.

This pipeline achieved promising results, having been able to predict the evolution of the robot's θ, x and y coordinates, from information regarding its surroundings through laser and odometry data, providing a new approach to this problem. This opens the possibility of other NMT-related architectures having the potential to be transformed and used in this type of problem.

A. Future Work

In this work, a first attempt at using NMT-related techniques to the robot pose estimation problem was implemented. Taking into account the achieved results by separating into two sub-problems, in future work, an overall solution for both orientation and position together could be tested. Additionally, other NMT-inspired solutions should be adapted into the robot's spacial localization paradigm, such as incorporating the use of GRU (Gated Recurrent Unit) cells instead or in conjugation with LSTM cells. Other enhancements that are possible to implement on these networks would be the addition of Attention and Bidirectional mechanisms.

Another interesting approach would be to see the results of Transformers, usually allowing for better process parallelization, reducing training and computing times, when compared to the general use of Seq2Seq architectures. The application of these approaches in future work and their behavior, when used in this new context, would allow for a better understanding of the application of these techniques in the SLAM domain.

References

[1] C. Breazeal, A. Takanishi, and T. Kobayashi, *Social Robots that Interact with People*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1349–1369. [Online]. Available: https://doi.org/10.1007/978-3-540-30301-5$_5$9

[2] C. M. Paulo Alvito, J. E. Paulo Carriço (IDM) Marco Barbosa, and D. A. S. D. G. (YDR), "MOnarch Robots Hardware D2.1.1," 2014.

[3] A. R. Khairuddin, M. S. Talib, and H. Haron, "Review on simultaneous localization and mapping (SLAM)," in *2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, 2015, pp. 85–90.

[4] H. Quan, Y. Li, and Y. Zhang, "A novel mobile robot navigation method based on deep reinforcement learning," *International Journal of Advanced Robotic Systems*, vol. 17, no. 3, pp. 1–11, 2020.

[5] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss, "OverlapNet: Loop Closing for LiDAR-based SLAM," no. i, 2020.

[6] L. Xiao, J. Wang, X. Qiu, Z. Rong, and X. Zou, "Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment," *Robotics and Autonomous Systems*, vol. 117, pp. 1–16, 2019. [Online]. Available: https://doi.org/10.1016/j.robot.2019.03.012

[7] T. Zheng, Z. Duan, J. Wang, G. Lu, S. Li, and Z. Yu, "Research on distance transform and neural network lidar information sampling classification-based semantic segmentation of 2d indoor room maps," *Sensors*, vol. 21, p. 1365, 02 2021.

[8] Q. L. Li, Y. Song, and Z. G. Hou, "Neural network based FastSLAM for autonomous robots in unknown environments," *Neurocomputing*, vol. 165, pp. 99–110, 2015.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," 2014.

[10] F. Suárez Bonilla and F. Ruiz Ugalde, "Automatic translation of spanish natural language commands to control robot comands based on lstm neural network," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 2019, pp. 125–131.

[11] "Recurrent neural networks," https://www.ibm.com/cloud/learn/recurrent-neural-networks, accessed: 2021-04-05.

[12] G. Tiwari, A. Sharma, A. Sahotra, and R. Kapoor, "English - Hindi Neural Machine Translation -," *International Conference on Communication and Signal Processing, July 28 - 30, 2020, India*, pp. 871–875, 2020.

[13] "Understanding LSTM Networks," https://colah.github.io/posts/2015-08-Understanding-LSTMs/, accessed: 2021-04-07.

[14] M.-t. Luong and C. D. Manning, "Stanford Neural Machine Translation Systems for Spoken Language Domains," *Iwslt-2015*, pp. 76–79, 2015.

[15] M.-T. Luong, "Neural machine translation. a dissertation submitted to department of computer science and the committee on graduate studies on Stanford University," December 2016.

[16] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015.

[17] A. Khan and A. Sarfaraz, "RNN-LSTM-GRU based language transformation," *Soft Computing*, vol. 23, no. 24, pp. 13 007–13 024, 2019.

[18] P. B. Anoop Kunchukuttan, Pratik Mehta, "The IIT Bombay English-Hindi Parallel Corpus. Language Resources and Evaluation Conference." 2018.

[19] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, "A Convolutional Encoder Model for Neural Machine Translation," *CoRR*, vol. abs/1611.02344, 2016.