A Collision Avoidance Method for Autonomous Underwater Vehicles Based on Long Short-Term Memories

László Antal[1(✉)], Martin Aubard[2], Erika Ábrahám[1], Ana Madureira[3], Luís Madureira[2], Maria Costa[2], José Pinto[2], and Renato Campos[2]

[1] RWTH Aachen University, Aachen, Germany
{antal,abraham}@cs.rwth-aachen.de
[2] OceanScan - Marine Systems and Technology, Lda., Porto, Portugal
{maubard,lmad,mariacosta,zepinto,rcampos}@oceanscan-mst.com
[3] Institute of Engineering, Polytechnic of Porto, Porto, Portugal
amd@isep.ipp.pt

Abstract. Over the past decades, underwater robotics has enjoyed growing popularity and relevance. While performing a mission, one crucial task for Autonomous Underwater Vehicles (AUVs) is *bottom tracking*, which should keep a constant distance from the seabed. Since static obstacles like walls, rocks, or shipwrecks can lie on the sea bottom, bottom tracking needs to be extended with *obstacle avoidance*. As AUVs face a wide range of uncertainties, implementing these essential operations is still challenging.

A simple *rule-based control* method has been proposed in [7] to realize obstacle avoidance. In this work, we propose an alternative *AI-based control* method using a *Long Short-Term Memory* network. We compare the performance of both methods using real-world data as well as via a simulator.

Keywords: Autonomous underwater vehicles · Obstacle avoidance · Rule-based control · AI-based control · Long short-term memories

1 Introduction

Autonomous Underwater Vehicles (AUVs) face a wide range of complex tasks in the underwater environment. Two of these tasks are *bottom tracking* and *obstacle avoidance*. Bottom tracking means that an underwater vehicle needs to maintain a distance to the seafloor as constant as possible. It helps the AUV to gather different types of sensor data more reliably (e.g., side-scan sonars, multibeam sonars, and camera images). However, since the seafloor's surface is uneven and it may happen that some obstacles (rocks, walls, or other static objects) are lying underneath, bottom tracking needs to be extended with obstacle avoidance.

In this paper, we consider Light Autonomous Underwater Vehicles (LAUVs) with restricted sensor information. The starting point for our work is a rule-based

A. Abraham et al. (Eds.): IBICA 2022, LNNS 649, pp. 448–457, 2023.
https://doi.org/10.1007/978-3-031-27499-2_42

control mechanism [7], which is the method currently running on LAUVs. The problem with this method is that it is sensible to noise, therefore, the decisions are not always reliable. In order to provide more robust decisions, we propose as an alternative approach an AI-based control method. We comparatively evaluate both approaches using the data gathered from real-time missions as well as using a simulator.

The rest of this paper is structured as follows. We discuss related work in Sect. 2 before we present the rule-based and the AI-based controllers in Sects. 3 and 4, respectively. In Sect. 5, we show and analyze the results of our experiments. Finally, Sect. 6 summarizes this work and lists some of the aspects concerned by future work.

2 Related Work

While submerged, an Autonomous Underwater Vehicle (AUV) can not communicate and be controlled by an operator. Thus, the AUV needs to understand its environment to act accordingly. To understand its environment, the AUV uses different types of sensors that provide information about the distance from the bottom (altitude), the surface (depth), and the front of the AUV. Most AUVs use multibeam echo sounder to detect objects in front of the AUV [15]; multibeam provides 2D horizontal information about the potential object detected. The AUV can find a safe horizontal path by processing the data to avoid the possible object [4].

In this work, we consider Light Autonomous Underwater Vehicles (LAUV) [1], which use a single-beam echo sounder. This sonar gives a single distance value from the AUV to the object and does not provide any information about the potential safety horizontal path to avoid it. In [3], the authors propose an intelligent single-beam echo sounder to avoid collisions using hybrid automata modeling. As explained later in Sect. 4, our novel AI-based controller should improve obstacle avoidance and bottom tracking from [7], where the control depends on the system's current state, in contrast to our AI-based controller, which exploits also information from the past (time series).

Machine Learning (*ML*) enjoys growing attraction in many fields. Computer vision based on ML has achieved great success in several tasks, such as classification [8,12], object detection [9], and segmentation [9]. These techniques have also been implemented into robots and AUVs [2] to improve the knowledge of the robot about its environment. Techniques such as Long-Short Term Memory (LSTM) [5] and Recurrent Neural Networks (RNN) [6] provide an accurate prediction based on time series and sequential learning, which is widely used in speech recognition and machine translation. Several works have been conducted to predict a robot's position or behavior based on these ML time series methods. In [11], the authors propose an RNN method to predict the relative horizontal velocities of an AUV using data from an Inertial Measurement Unit (IMU), pressure sensor, and control inputs for dead-reckoning navigation. Thanks to the promising results based on the RNN implementation into AUV navigation,

the work [16] proposed a deep framework called NavNet by taking AUV navigation as a deep sequential learning problem. However, a typical RNN can face many challenges due to its limitations in memorizing long data sequences, which can affect the past time window used to predict the AUV behavior. The LSTM method, based on long and short-term memory, outperforms the RNN characteristic [13]. In [14], the authors implement an LTSM-based Dead Reckoning approach to estimate the surge and sway body-fixed frame velocities when the AUV is submerged.

3 Rule-based Control

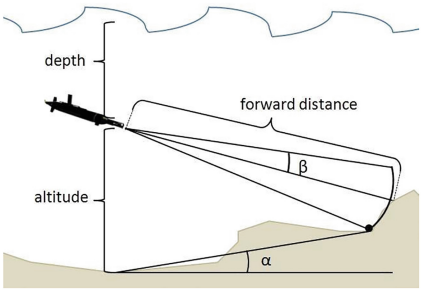

Fig. 1. Visualization of the sensor measurements considered in [7].

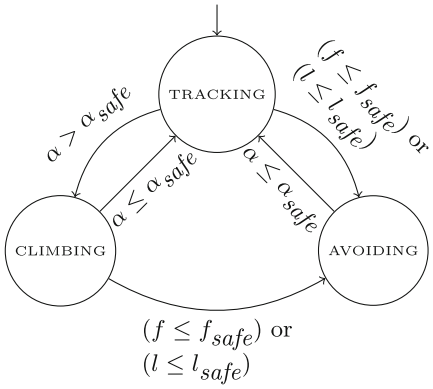

Fig. 2. The corresponding finite state machine of the method.

In order to tackle the obstacle-avoidance problem, the authors of [7] implement a very simple, yet very effective, rule-based approach. This method works in a reactive way, i.e., at every control cycle (timestamp), it considers the current measurements from specific sensors, and using simple trigonometry, it calculates the steepness of the sea bottom. The sensor values taken into consideration are as follows (see Fig. 1 for further details):

1. Depth measurement d: the vertical distance of the AUV to the water surface obtained using the depth sensor.
2. Altitude measurement l: the vertical distance of the AUV to the sea bottom, obtained using the multibeam DVL sensor.
3. Forward distance measurement f: the distance to a detected object in the facing direction of the AUV. These values are provided by a single-beam echo sounder sensor, considering a non-zero aperture β of the beam (see Fig. 1).

Based on these three measurements coming from the sensors, the method calculates the steepness α of the sea bottom, and it tries to adapt the pitch

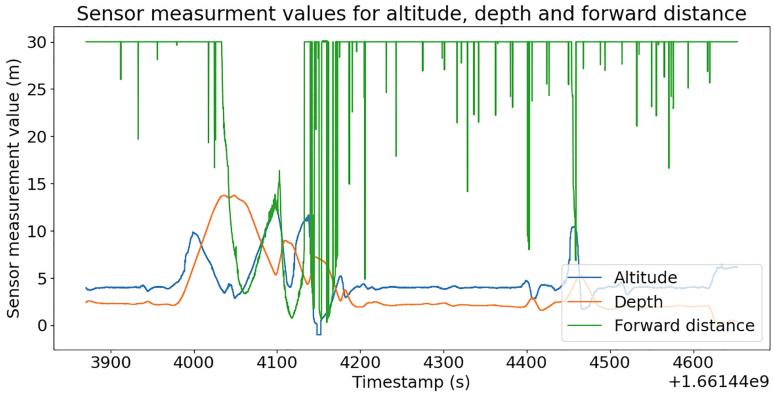

Fig. 3. Illustration of the noise in sensor measurements during a real-world mission. The altitude and depth values coming from the DVL and depth sensors are relatively robust, but the forward distance measurements of the echo sounder are very noisy and, therefore, unsuitable for reactive obstacle avoidance.

using a finite state machine (see Fig. 2) with three states: *tracking* (when the AUV tries to maintain constant altitude), *climbing* (when the seafloor is too steep and the vehicle is pitching up), *avoiding* (when the distance to the object or to the seafloor is too short and the AUV stops the thruster). If the vehicle is in the *avoiding* state, since the thruster is stopped, the buoyancy pulls the AUV upward until the obstacle in front "disappears" from the echo sounder's field of view. When that happens, the vehicle goes back to the *tracking* state.

This simple rule-based control can already fulfill the bottom-tracking task, but it has some limitations. (i) Due to the imprecise, noisy aspect of the measured sensor values, the method lacks robustness in some cases. In order to get an impression, Fig. 3 shows the sensor measurements during a real mission made by the OceanScan MST company with a LAUV at Matosinhos harbor, Portugal. (ii) This controller considers only the current sensor values and ignores the past time frame. (iii) Finally, the rules operate with hard pre-defined threshold values (α_{safe}, l_{safe}, f_{safe}), though it is possible that for different AUVs or different environments, fine-tuning of the threshold values would be necessary.

4 AI-based Control

To solve the problems mentioned in Sect. 3, we propose a machine-learning-based approach. As one subclass of recurrent neural networks, Long Short-Term Memories (LSTM) can handle well time-series data and long-term time dependencies [5]. The idea is to take fixed-length time windows containing the consecutive sensor measurements from the near past and use time-series classification.

Our aim is to learn for a given AUV time series the correct maneuver, which is one of the AUV states *tracking*, *climbing*, and *avoiding* extended with two

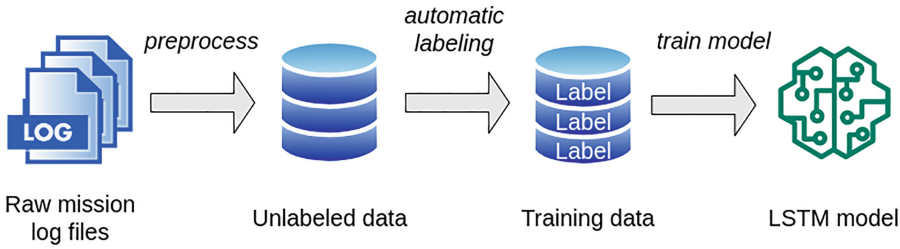

Fig. 4. The proposed pipeline to gather, preprocess and label training data from the log files of real missions and to train the neural network controller.

auxiliary states *unsteady* and *surfaced*. The *unsteady* state is triggered when the available data is too noisy and, therefore, we cannot make a reliable decision; in this case, continuing the previous maneuver or trying to stabilize the AUV would be a proper action. The *surfaced* state is triggered when the vehicle is on the surface. The reason why it is necessary to distinguish this state is that the echo sounder sensor does not work on the surface. Thus, we do not have any information about a potential obstacle in front of the AUV, so a different type of control is needed.

The LSTM network should output the state that the AUV should enter in order to circumvent collisions. We expect that the model learns to make reliable predictions even when the data is noisy. Furthermore, we expect it also to generalize better to different settings (i.e., different AUVs or environments) than the original rule-based approach [7].

To train the LSTM, first, we need to acquire the necessary training, validation and test data. This process happens in three steps, which we visualize in Fig. 4. The steps involved in the pipeline bring up the following essential questions that we answer partly in this section and partly in Sect. 5.

The first question is how we can produce the time windows containing the sensor measurements coupled with the correct classification label. Since generating data from simulation would not result in realistic scenarios, we considered log files of real missions in this paper. These missions were executed by the OceanScan MST company using a LAUV. Using the log files, we extracted the necessary sensor data in CSV format from Neptus [10], which is the command and monitor software for the LAUV.

Secondly, each time series data gathered from the log files needs to be labeled with a suitable output that defines one of the five states that need to be activated in order to avoid a collision. Manual labeling would not be feasible since it takes a lot of time and effort, and the result may not be as precise as we want. Consequently, we developed an automatic method for labeling the time-series data. We describe this method in Sect. 4.1.

Finally, we train an LSTM network using the automatically labeled training data. We describe the parameter settings for the training process in Sect. 5.

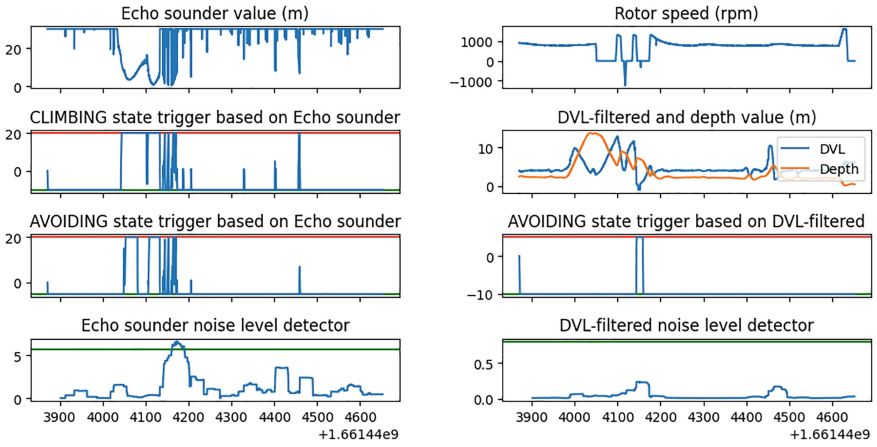

Fig. 5. Visualization of the automatic labeling method applied for the same mission as the one shown in Fig. 3. The subplots *Echo sounder value*, *Rotor speed*, and *DVL-filtered and depth value* are the corresponding plots for the relevant sensor values. The *CLIMBING/AVOIDING state triggers* are the noise gates applied on the sensor values with a different attack (red line) or release time (green line). Lastly, the *noise level detectors* are plotted in the last row, and the green line shows the noise indicator threshold.

4.1 Automatic Labeling of the Data

After gathering all the raw, unlabeled data from the mission log files, we consider the data as a set of multi-dimensional (multi-sensor) time series. The task is to assign a label, one of the five possible maneuvers (*tracking*, *climbing*, *avoiding*, *unsteady*, and *surfaced*), to each timestamp, taking into account the data at the current timestamp and the data series measured before the current timestamp.

The *unsteady* state should be triggered when the data is too noisy so that we can make no reliable decision. For a given timestamp, we define its *noise level* as the standard deviation of the first-order discrete difference (i.e., of the absolute values of the differences between the successive sensor values) over the considered time window up to the current timestamp (with the same size as the input for the neural network). We label those timestamps as *unsteady*, whose noise level exceeds a certain *noise indicator threshold* value, as illustrated in the bottom two subplots of Fig. 5.

The *climbing* state is triggered using a *noise gate* with three parameters: an initial value i, a release time r, and an attack time a with $r \leq i \leq a$. We initialize a counter with the initial value i and update it iteratively for each timestamp in chronological order as follows: In case the rule-based controller would choose the *climbing* state, then (i) if the counter value equals the release time, then we set it to the initial value and (ii) if the counter value is below the attack time then we increase it by one. Otherwise, if the rule-based controller would *not* choose the *climbing* state, then (i) if the counter value equals the attack time, then we set it to the initial value, and (ii) if the counter value is above the release time, then we decrease it by one. After these calculations, we label the data at

Table 1. The parameter settings for the state triggers (first four rows) and the noise detectors (last two rows).

Measured sensor type	Triggered state	Threshold value	Attack time	Release time
Echo sounder	*climbing*	15 m	20	10
Echo sounder	*avoiding*	8 m	20	5
DVL-filtered	*avoiding*	1.2 m	5	10
Depth sensor	*surfaced*	0.5 m	5	5
Echo sounder	*unsteady*	5.75	–	–
DVL-filtered	*unsteady*	0.80	–	–

the current timestamp with the command *climbing* if the attack time has been reached at least once and the release time has not been reached after the last such occurrence. With the three parameters, we are able to tune the sensitivity of the trigger to enter and exit the climbing state.

Labeling with the *avoiding* and *surfaced* states is analogous to *climbing*. The trigger for the *surfaced* state is not plotted in Fig. 5 because it is not interesting for this mission, in which surfacing does not happen. Nonetheless, it would be computed in a very similar way as the *climbing* and *avoiding* triggers.

Lastly, the *tracking* is the selected label in case there are no other assigned labels. The analysis of the automatic labeling method for a subset of the possible labels is shown in Fig. 5. In case a timestamp gets multiple labels, we consider the following priority: $unsteady > surfaced > avoiding > (tracking \,|\, climbing)$.

5 Experimental Results and Their Evaluation

In this section, we first present the hyperparameter values/settings used to do the experiments, before we report on the experimental results and analyze them.

For the automatic labeling process, we summarize the state trigger's parameter values and the noise detector's threshold values in Table 1. Their values were empirically determined.

We applied the labeling using the listed parameter values, with a sliding window size of 300 timestamps corresponding to a 1-minute timespan considering the usual 5 Hz sampling rate. The size is the same as the input size of the LSTM, and the reason we chose it is that the window size needs to be the smallest possible, for which a label could be assigned unambiguously. Since accurate data should safely determine the actual label, only the noise could cause mislabeling. The window size of 300 is enough because it is very unlikely that a time window would contain this amount of false or noisy data. The result of the labeling process is illustrated in Fig. 6.

For the neural network training, we used 17 log files as training and validation data, achieving a validation accuracy of 98.93%. We tested the model with the remaining 13 mission files, achieving an average accuracy of 97.35%. The high accuracy of the model indicates that it learned well to give the same results as

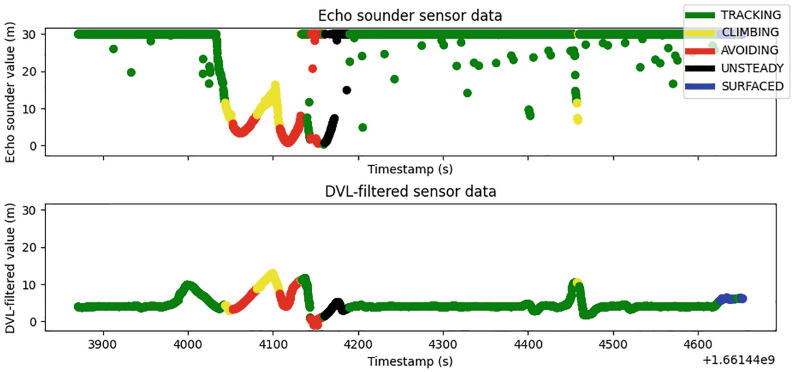

Fig. 6. The result of the automatic labeling process. Two sensor measurements are plotted, and each data point has the color of the corresponding state.

the automatic labeling method. The advantage of using a neural network is that it can learn the data from multiple differently parametrized labeling processes so that the neural network can learn to *generalize* the predictions for different AUVs or environments. We train the model using ten epochs with a batch size of 64. For the structure of the neural network, there is a wide range of choices. We used the following parameter values, but mention that experimenting with different architectures could further improve the performance:

- Convolutional layer (1D): 32 filters, kernel size of 3, ReLU activation function
- Max-pooling layer (1D): pool size of 2
- LSTM layer: 256 LSTM cells
- Fully-connected layer: 5 units, softmax activation function

After training, the model was tested in a simulated environment using Dune and Neptus. In order to run the simulator realistically, it needs the bathymetry measurements. Unfortunately, we had only a small region of Porto's harbor's bathymetry mapped in the simulator, so we conducted one test survey using this mapped region. During the mission, the AUV has to climb on a wall twice.

We show the comparison of the original rule-based control and the neural network control in Fig. 7. The green rectangles (solid) show when the neural network controller sends control maneuvers to avoid the wall. The red rectangles show the timestamps where the rule-base controller avoids the wall; the green dashed rectangles are shown just for comparison. It is observable that using the neural network controller, the AUV starts climbing on the wall, passes it earlier, and finishes the entire mission sooner than the rule-based control. It is worth mentioning that the AUV has a limit of 15° for the maximum pitch, which means that if a wall is steeper than 15°, then the AUV will not be able to avoid the wall only using the *climbing* maneuver. In this particular scenario, the *avoiding* state will be triggered.

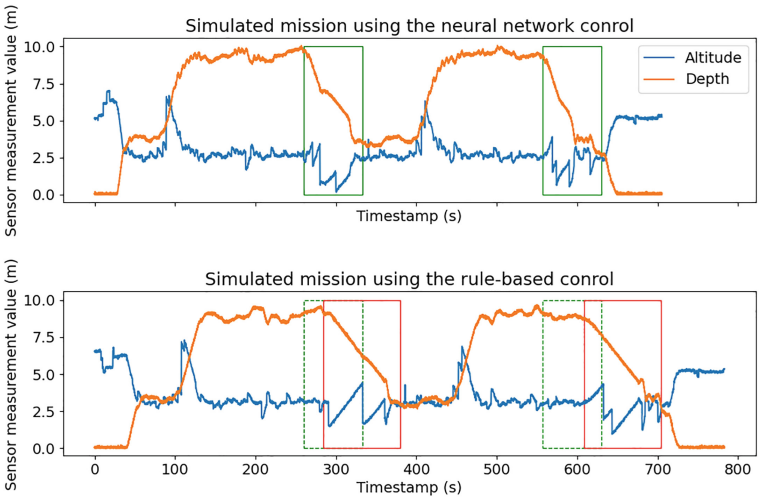

Fig. 7. Comparison of the neural network and the original rule-based control method.

6 Conclusion and Future Work

In this paper we proposed a pipeline to train a neural network that manages obstacle avoidance. The pipeline consists of multiple steps. First, we gathered the sensor data from multiple log files recorded during different missions. In order to use the raw data for training a model, we presented an automatic labeling method. With the labeling method, we assign a state (one of the five possible maneuvers) to each timestamp of the set of time-series data. Finally, using the labeled data, we trained a Long Short-Term Memory network and tested it in a simulator environment. In Sect. 5, we show the parameters and settings used to do the experiments. Regarding the future, our plan is first to extend the simulator's bathymetry map such that we can execute more simulated tests. Furthermore, we intend to deploy and test the neural network on a real AUV. If needed, we will use more training data and fine-tune the model's parameters. With the neural network controller, we aim to increase the efficiency of the bottom-tracking algorithm. The efficiency can be defined multiple ways, however, our desire is the following:

- We want to reduce the overall mission time with more efficient wall climbing.
- Also, we would like to keep a constant altitude whenever possible, such that the collected data will have better quality.
- Finally, using the LSTM, we would like to investigate the problem of predicting the presence of a wall close to the AUV but not observable by the sensor measurements yet.

Acknowledgements. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 956200. For more info, please visit https://remaro.eu.

References

1. Alexandre, S., et al.: Lauv: The man-portable autonomous underwater vehicle. In: IFAC Proceedings (2012)
2. Aubard, M., Madureira, A., Madureira, L., Pinto, J.: Real-time automatic wall detection and localization based on side scan sonar images. In: IEEE (2022)
3. Calado, P., et al.: Obstacle avoidance using echo sounder sonar. In: OCEANS 2011 IEEE-Spain, pp. 1–6. IEEE (2011)
4. Healey, A.J.: Obstacle avoidance while bottom following for the Remus autonomous underwater vehicle. IFAC Proceedings Volumes **37**(8), 251–256 (2004)
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
6. Jordan, M.I.: Serial order: a parallel distributed processing approach. In: Advances in psychology, vol. 121, pp. 471–495. Elsevier (1997)
7. Madureira, L., et al.: The light autonomous underwater vehicle: evolutions and networking. In: 2013 MTS/IEEE OCEANS-Bergen. pp. 1–6. IEEE (2013)
8. Nayak, N., Nara, M., Gambin, T., Wood, Z., Clark, C.M.: Machine learning techniques for AUV side-scan sonar data feature extraction as applied to intelligent search for underwater archaeological sites. In: Field and Service Robotics (2021)
9. Neves, G., Ruiz, M., Fontinele, J., Oliveira, L.: Rotated object detection with forward-looking sonar in underwater applications. Expert Syst. Appl. **140**, 112870 (2020)
10. Pinto, J., Dias, P.S., Martins, R., Fortuna, J., Marques, E., Sousa, J.: The LSTS toolchain for networked vehicle systems. In: 2013 MTS/IEEE OCEANS-Bergen, pp. 1–9. IEEE (2013)
11. Saksvik, I.B., Alcocer, A., Hassani, V.: A deep learning approach to dead-reckoning navigation for autonomous underwater vehicles with limited sensor payloads. In: OCEANS 2021: San Diego–Porto. pp. 1–9. IEEE (2021)
12. Samaras, S., et al.: Deep learning on multi sensor data for counter UAV applications-a systematic review. Sensors **19**(22), 4837 (2019)
13. Sherstinsky, A.: Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, p. 132306 (2020)
14. Topini, E., et al.: LSTM-based dead reckoning navigation for autonomous underwater vehicles. In: Global Oceans 2020: Singapore–US Gulf Coast. pp. 1–7. IEEE (2020)
15. Yan, Z., Li, J., Jiang, A., Wang, L.: An obstacle avoidance algorithm for AUV based on obstacle's detected outline. In: 2018 37th Chinese Control Conference (CCC), pp. 5257–5262. IEEE (2018)
16. Zhang, X., He, B., Li, G., Mu, X., Zhou, Y., Mang, T.: Navnet: AUV navigation through deep sequential learning. IEEE Access **8**, 59845–59861 (2020)