Bioinspired SLAM: A New Implementation of NeoSLAM and a Comparative Evaluation with RatSLAM

***Abstract*— This paper presents a new implementation of the NeoSLAM algorithm. The proposed version is a complete rewrite of NeoSLAM into a modular architecture using modern frameworks and introduces additional parameters that increase robustness. This work also provides a comparative evaluation between NeoSLAM and RatSLAM using four datasets under different environmental conditions. Experimental results highlight differences in mapping consistency and trajectory reconstruction, demonstrating the effectiveness and practical applicability of the proposed ROS 2–based version. The results indicate that NeoSLAM achieves performance comparable to RatSLAM, with a slight advantage in the evaluated datasets.**

I. Introduction

Autonomous mobile robotics faces substantial challenges in real-world applications, particularly when operating in unknown, dynamic, or GPS-denied environments. Such scenarios, ranging from subterranean exploration to forest environments and underwater navigation, demand reliable and accurate localization and mapping capabilities [1]. In these scenarios, robots must operate without prior maps while coping with perceptual aliasing, sensor noise, and environmental changes. Reliable localization and mapping are therefore critical to ensure safe and efficient operation, especially in long-term deployments where uncertainty accumulates over time [2].

Simultaneous Localization and Mapping (SLAM) addresses this challenge by enabling a robot to build a map while estimating its own pose within that map. By fusing sensory observations with motion models, SLAM algorithms iteratively reduce state uncertainty and maintain global consistency. Loop closure detection further mitigates accumulated drift and plays a key role in long-term autonomy and large-scale exploration [2], [3].

Among the various SLAM paradigms, biologically inspired approaches have gained attention, beginning with RatSLAM, which draws inspiration from the rodent hippocampal navigation system [4], [5]. Subsequent models incorporated findings related to place cells, grid cells, and head direction cells to improve spatial representation and robustness [6]–[9]. Compared to classical probabilistic SLAM methods, these approaches often provide scalable topological representations and improved tolerance to perceptual ambiguity in visually complex environments, allowing them to operate in scenarios where traditional probabilistic methods often fail completely [5].

Building upon these foundations, NeoSLAM was recently proposed as a neuroscience-based model motivated by advances in understanding the neocortex and hippocampal–entorhinal circuits [10]. Unlike systems primarily inspired by hippocampal mechanisms and typically modeled using a Continuous Attractor Neural Network (CANN), NeoSLAM also integrates neocortical principles, including a sequence memory model based on Hierarchical Temporal Memory (HTM) [11] and Sparse Distributed Representations (SDRs) [12]. This hybrid design enables improved sequence learning, adaptability to appearance changes, and enhanced long-term visual place recognition performance under varying environmental conditions.

The original NeoSLAM [1] implementation was developed within the Robot Operating System (ROS) 1 workspace. Although functional, this implementation presents architectural limitations that create bottlenecks in real-time data processing, preventing the system from handling all incoming sensory data at full rate. Furthermore, the modeling of SDRs and HTM relied on the `NuPIC` [2] legacy framework, implemented in Python 2.7. This dependency constrained the entire software stack to outdated technologies, including ROS 1 Melodic and Ubuntu 18.04, limiting maintainability, scalability, and long-term support.

This new implementation of NeoSLAM [3] overcomes the aforementioned limitations. In summary, the contributions of this work are as follows:

- **Architectural Refactoring:** The original codebase was fully refactored into a modular and extensible architecture, with a redesigned execution pipeline that improves computational efficiency and prevents data loss during real-time operation.
- **New Tuning Parameters:** Additional parameters were introduced to enhance robustness. These include image cropping mechanisms to remove irrelevant or harmful regions from the input (as occurs in RatSLAM) and a control parameter to avoid processing temporally redundant frames.
- **Integration of the modern HTM framework:** The legacy `NuPIC` framework in Python 2.7 was replaced by the modern `htm.core` [4] library, implemented in C++17 with Python 3 bindings, ensuring improved performance and maintainability.
- **Migration to ROS 2:** The adoption of `htm.core` enabled the transition to a ROS 2 Rolling-based architecture, leveraging modern middleware capabilities, improved communication mechanisms, and enhanced

[1] https://github.com/cappizzino/neoslam_ws
[2] https://github.com/numenta/nupic-legacy
[3] https://github.com/NeuroscienceRobotics/neoslam
[4] https://github.com/htm-community/htm.core

scalability.
- **Experimental Evaluation:** NeoSLAM is systematically evaluated against RatSLAM across four distinct datasets, providing a comprehensive comparative analysis.

II. System Overview

NeoSLAM comprises seven main modules: Visual Feature Extractor, Binary Projector, Neocortex, Spatial-View Cells, Pose Cells, Experience Map and Visual Odometry. Figure 1 illustrates the main modules described as ROS 2 nodes, along with the corresponding ROS topics describing the data flow between them.

A. Visual Feature Extractor Node

The original NeoSLAM implementation exhibited processing bottlenecks during real-time operation, in which the system was unable to process every incoming frame depending on the image publication frequency. This resulted in non-deterministic frame loss and performance variations dependent on the available computational resources. However, this intermittent processing often proved beneficial, as temporally redundant frames were implicitly skipped. To reproduce this behavior in a controlled and deterministic manner, this version introduces the `frame_stride` parameter $s \in \mathbb{N}$. Given a sequence of input images, it specifies that only one out of every s frames is processed, while the remaining $s-1$ frames are ignored, effectively performing uniform temporal downsampling prior to feature extraction.

Furthermore, cropping parameters were introduced in the pre-processing stage to restrict the analysis to relevant visual regions. Let the downsampled input image be represented as $I \in \mathbb{R}^{W \times H}$, where W and H denote width and height, respectively. When `crop_image` is enabled, a subregion is defined by horizontal limits $[x_{\min}, x_{\max})$ and vertical limits $[y_{\min}, y_{\max})$, satisfying $0 \leq x_{\min} < x_{\max} \leq W$ and $0 \leq y_{\min} < y_{\max} \leq H$. The cropped image is then given by

$$I_{\text{crop}}(x', y') = I(x' + x_{\min},\, y' + y_{\min}), \tag{1}$$

for $0 \leq x' < x_{\max} - x_{\min}$ and $0 \leq y' < y_{\max} - y_{\min}$. This operation ensures that feature extraction is restricted to the selected region of interest.

Subsequently, a pre-trained *AlexNet conv3* feature extractor [10], [14] produces the descriptor $\mathbf{f} = \text{CNN}(I_{crop})$ where $\mathbf{f} \in \mathbb{R}^n$ with $n = 64896$. The descriptor is published to the `/visual_features` topic.

B. Binary Projector Node

The Binary Projector Node is composed of two stages: dimensionality reduction and binarization.

First, the dimensionality reduction stage is performed using a random gaussian projection defined as $\mathbf{d} = P\mathbf{f}$, where $P \in \mathbb{R}^{m \times n}$ is the gaussian random projection matrix, and $\mathbf{d} \in \mathbb{R}^m$ is the reduced descriptor with $m = 1024$ dimensions, with $m < n$. This reduction decreases computational cost and memory usage while maintaining structural properties necessary for subsequent encoding [13].

In the second stage, the Sparse Locality Sensitive Binary Hashing (sLSBH) method [10], [14] performs binarization and sparsification to obtain a compact and robust binary representation. Given the projected descriptor $\mathbf{d} \in \mathbb{R}^m$, with $m = 1024$ in our configuration, the algorithm selects the $s = 25\%$ largest and $s = 25\%$ smallest components of $\mathbf{d}$ to construct a sparse binary code.

The final binary vector is defined as the concatenation

$$\mathbf{z} = \left[\mathbf{z}^+ \; \mathbf{z}^-\right]^T, \tag{2}$$

where $\mathbf{z}^+ \in \{0,1\}^m$ encodes the largest coefficients and $\mathbf{z}^- \in \{0,1\}^m$ encodes the smallest coefficients of $\mathbf{d}$. Since $m = 1024$ and $s = 25\%$, each vector contains $\frac{s}{100} \cdot m = 256$ active bits.

Therefore, each subvector $\mathbf{z}^+$ and $\mathbf{z}^-$ has length 1024 with 256 active bits, and their concatenation produces the final descriptor $\mathbf{z} \in \{0,1\}^{2m}$ of dimension 2048 with a total of 512 active bits. The $\mathbf{z}$ binary descriptor is written in the `/bin_features` topic.

C. Neocortex Node

The Neocortex Node transforms the reduced binary visual information into a format that captures spatial and temporal features grounded in the principles of *Hierarchical Temporal Memory* (HTM) [11]. To facilitate understanding of this module, the fundamental structures required for operating an HTM network are described: Sparse Distributed Representation (SDR), Spatial Pooler (SP), and Temporal Memory (TM).

1) Sparse Distributed Representation: An SDR is defined as an n-dimensional binary vector $\mathbf{x} = [b_1, \ldots, b_n]$, where $\mathbf{x} \in \{0,1\}^p$ and only a small fraction of components are active ($b_i = 1$).

The overlap Φ between two SDRs $\mathbf{x_1}$ and $\mathbf{x_2}$ measures similarity by counting shared active bits:

$$\Phi(\mathbf{x_1}, \mathbf{x_2}) = \mathbf{x_1} \cdot \mathbf{x_2} = \|\mathbf{x}_1 \cap \mathbf{x}_2\|_0. \tag{3}$$

A match Ψ occurs when the overlap exceeds a predefined threshold β:

$$\Psi = \Phi(\mathbf{x_1}, \mathbf{x_2}) \geq \beta. \tag{4}$$

2) Spatial Pooler: The Spatial Pooler (SP) converts inputs into SDRs, providing properties that are typically absent in raw inputs. By activating only a small fraction of minicolumns through a competitive mechanism, the SP enforces fixed sparsity and improves representational efficiency. At the same time, it increases robustness to noise by mapping similar inputs to overlapping sets of active columns, producing stable representations under small input perturbations [19]. In this work, the binary descriptor is directly converted to an SDR and the SP component is not used.

3) Temporal Memory: The Temporal Memory (TM) is a recurrent binary-state network that accumulates information over consecutive input SDRs in order to encode the temporal context of the current input. It operates on a layer composed of n minicolumns (C_j), each containing m cells ($c_{i,j}$), where contextual information is represented by distinct cells within

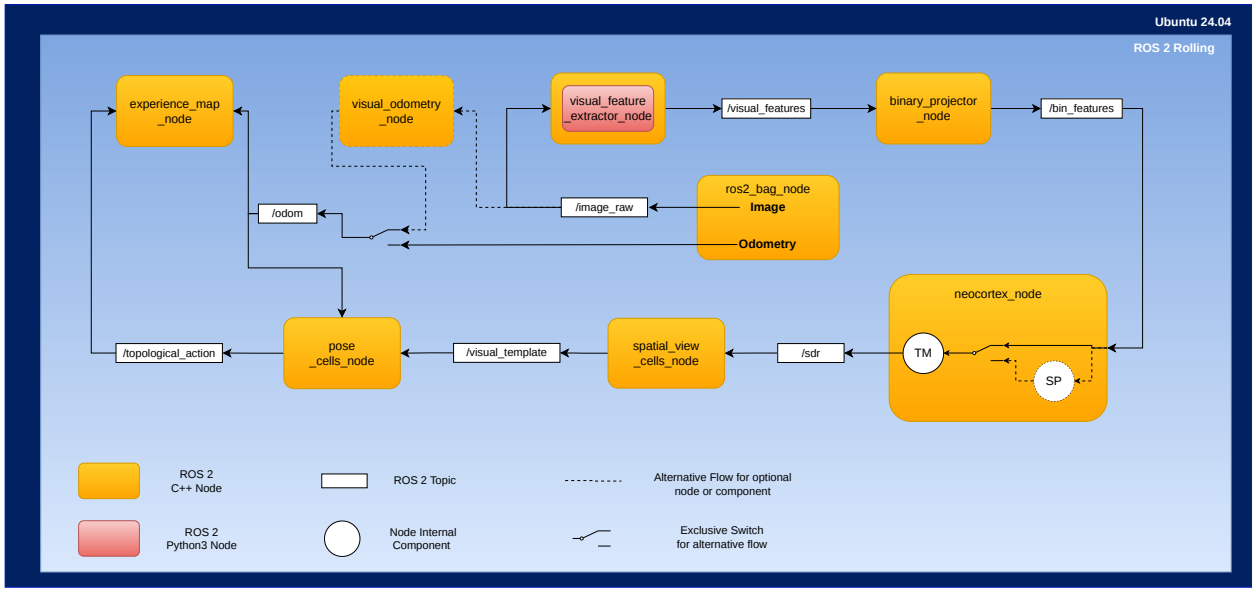

Fig. 1: NeoSLAM architecture: odometry and image data are acquired either directly from sensors or from a ROS 2 bag file, then processed by a sequence of nodes, flowing through the entire pipeline until the final experience map is generated.

active minicolumns. The TM receives as input a SDR and, at each time step $t \in \mathbb{N}_0$, its state is characterized by three binary variables per cell: active $a_{i,j}^t$, predictive (depolarized) $\pi_{i,j}^t$, and winner $w_{i,j}^t$, grouped into the matrices A^t, Π^t, and W^t, respectively.

The TM dynamics begin with the calculation of the active state of the cells according to the equation (Eq. 5):

$$a_{i,j}^t = 1 \iff A_j^t \wedge \left(\pi_{i,j}^{t-1} \vee \left(\forall m : \neg\pi_{m,j}^{t-1}\right)\right) \tag{5}$$

A cell is activated within a winning column ($A_j^t = 1$) if it was previously predicted ($\pi_{i,j}^{t-1} = 1$) or if no cells in that minicolumn were in a predictive state. In the latter case, a "bursting" phenomenon occurs, where all cells in the column are activated to represent a new context.

Let $D_{ij}^d = \{s_{ij}^d\}$ be the synaptic permanence matrix from a particular distal segment d where each synapse is bounded by $0 \leq s_{i,j}^d \leq 1$, and $\tilde{D}_{ij}^d = 1$ if $D_{ij}^d > \epsilon$ be the binary matrix of connected synapses for some permanence threshold. Then, the predictive state for the current time step t is determined by the equation 6:

$$\pi_{i,j}^t = 1 \iff \exists_k \left\| \tilde{D}_{i,j}^d \circ A^t \right\|_1 > \theta \tag{6}$$

where $\theta \in \mathbb{N}$ represents the dendritic segment activation threshold and $\circ$ denotes the element-wise multiplication (Schur product) . A cell becomes depolarized if at least one of its distal dendritic segments d has a sufficient number of connected synapses with currently active presynaptic cells.

Learning occurs through the selection of winner cells that reinforce sequence transitions. If a prediction is successful, the winner cell is identified by equation 7:

$$\begin{aligned} w_{i,j}^t = 1 \iff & A_j^t \wedge (\pi_{i,j}^{t-1} > 0) \\ & \wedge \|\tilde{D}_{i,j}^d \circ A^{t-1}\|_1 > \theta \end{aligned} \tag{7}$$

However, if bursting occurs, Equation 8 selects the cell that was closest to activation (the one with the highest L_1 norm), even if it was below the threshold θ, to represent the future context:

$$\begin{aligned} w_{i,j}^t = 1 \Leftrightarrow & A_j^t \wedge \left(\forall m : \neg\pi_{m,j}^{t-1}\right) \\ & \wedge \|\dot{D}_{i,j}^d \circ A^{t-1}\|_1 = \\ & \max_i \left(\|\dot{D}_{i,j}^d \circ A^{t-1}\|_1\right) \end{aligned} \tag{8}$$

where $\dot{\mathbf{D}}_{ij}^d = 1$ if $\mathbf{D}_{ij}^d > 0$, and 0 otherwise.

Finally, synaptic permanence values are adjusted using a Hebbian-like rule (Eq. 9):

$$\Delta D_{i,j}^d = \epsilon_i \left(\dot{D}_{i,j}^d \circ A^{t-1}\right) - \epsilon_d \dot{D}_{i,j}^d \tag{9}$$

where the factors ϵ_i and ϵ_d increase the permanence of synapses with active presynaptic cells and decrease it for those with inactive cells, respectively.

Now that the fundamental structures of HTM network were defined, the specific parameters implemented in this work are explained. The TM module inside the Neocortex node receives an input SDR $\mathbf{z} \in \{0,1\}^{2048}$ containing 512 active bits (activating 512 columns A_j^t). Through the HTM dynamics, this representation is expanded into a higher-dimensional SDR $\mathbf{y} \in \{0,1\}^{32 \cdot 2048} = \{0,1\}^{65536}$, corresponding to 32 cells per minicolumn. Each column represents distinct visual features and while all cells inside a column can be activated by the same SDR input, the temporal context distingshes them. Therefore, the resulting output SDR increases representational capacity, enabling robust sequence encoding and improved discrimination of temporally correlated visual patterns.

D. Spatial-View Cells Node

In rodents, navigation is largely described as occurring from place to place and relies primarily on local tactile and olfactory cues. In contrast, the highly developed visual system of primates enables the identification of locations directly from viewed scenes, a distinction that has important implications for how the primate, including the human hippocampus, supports episodic memory processes [18]. Inspired by Spatial-View cells found in primates, a computational model aimed at visual place recognition and loop closure detection was proposed [10], associating neocortical visual representations with previously stored visual experiences to recognize revisited locations. The Spatial-View Cells Node introduced in this work implements the same underlying model with improved computational efficiency and presents it using a complete mathematical formulation.

At each time step $t \in \mathbb{N}_0$, the system observes a SDR $\mathbf{y}_t \in \{0,1\}^{65536}$ generated by the neocortical model, with 512 active bits. Those observations are grouped into intervals $I_k = [t_k^{start}, t_k^{end}]$, where k denotes the interval index and $\rho_k = t_k^{end} - t_k^{start} \geq 1$ the interval size. Each interval maintains an accumulated SDR representation defined as the union of all SDRs within the interval

$$\mathbf{Y}_k = \bigcup_{t=t_k^{start}}^{t_k^{end}} \mathbf{y}_t. \tag{10}$$

Given a new observation $\mathbf{y}_t$, two parameters define if the observation will join the current interval or create a new one: the similarity threshold $\theta_\alpha \in \mathbb{N}^+$ for interval membership (`theta_alpha`) and the maximum interval size threshold $\theta_\rho \in \mathbb{N}^+$ (`theta_rho`). If the similarity (which corresponds to the overlap equation 3) between the current interval k_{curr}, computed as

$$\alpha(t) = \mathbf{Y}_{k_{curr}} \cdot \mathbf{y}_t = \|\mathbf{Y}_{k_{curr}} \cap \mathbf{y}_t\|_0 \tag{11}$$

is high enough $\alpha(t) \geq \theta_\alpha$ and does not extrapolate the maximum interval size $\rho_{k_{curr}} < \theta_\rho$, then the interval is extended $\mathbf{Y}_{k_{curr}} \leftarrow \mathbf{Y}_{k_{curr}} \cup \mathbf{y}_t$; otherwise, a new interval is created and its representation is initialized as $\mathbf{Y}_{k+1} = \mathbf{y}_t$. All interval representations are stored in the matrix $\mathbf{M}_{intervals}^\top = \begin{bmatrix}\mathbf{Y}_1^\top & \mathbf{Y}_2^\top & \cdots & \mathbf{Y}_K^\top\end{bmatrix}$ where K is the total number of intervals.

Loop closures are detected by computing the similarity between the current observation and all intervals, producing the vector $\mathbf{s}(t) = [s_1(t), \ldots, s_K(t)]^T$, where $s_k(t) = \mathbf{Y}_k \cdot \mathbf{y}_t = \|\mathbf{Y}_k \cap \mathbf{y}_t\|_0$. Here, two other parameters are important for considering loop closures: the loop closure threshold $\theta_\sigma \in \mathbb{N}^+$ used to define the loop closure occurrence and the new proposed parameter `exclude_recent_intervals` defined as $\omega \in \mathbb{N}_0$ used to avoid trivial matches with recent observations. Recent intervals are excluded by considering only the first $K-\omega$ elements of $\mathbf{s}(t)$. Candidate loop closures are then defined as $\mathcal{M}(t) = \{k \in \{1, \ldots, K-\omega\} : s_k(t) > \theta_\sigma\}$. If $\mathcal{M}(t) \neq \emptyset$, the best matching interval is selected as $k^* = \arg\max_k s_k(t)$. Each interval is associated with a visual template identifier through the mapping $\phi : \{1, \ldots, K\} \rightarrow \mathbb{N}$. The template assigned to the current observation is

$$\tau(t) = \begin{cases} \phi(k^*) & \text{if } \mathcal{M}(t) \neq \emptyset \\ \tau_{curr} & \text{if } \mathcal{M}(t) = \emptyset \wedge k_{prev} = k_{curr} \\ \tau_{next} & \text{if } \mathcal{M}(t) = \emptyset \wedge k_{prev} \neq k_{curr} \end{cases} \tag{12}$$

Thus, loop closures reuse the template of the most similar interval, while new intervals without matches generate new template identifiers. The template message is published through the `/visual_template` topic.

E. Pose Cells Node

Pose cells simulate neurons in the hippocampal and entorhinal cortex to provide spatial representation and navigation. This 3D continuous attractor network emulates the cognitive mapping of a rodent by encoding the pose of the robot (x, y, θ) as it explores its environment [17].

As the robot moves (reading from the `/odom` topic), pose cells fire to create neural signatures associated with specific locations (from the `/visual_template` topic). By integrating these activations with motion information over time, the pose cell network estimates the robot's position and determines whether the topological graph must be updated. The message to update is published through the `/topological_action` topic and includes the action type, the IDs of the involved experiences (source and destination), and their relative orientation. The whole model is described in [15].

F. Experience Map Node

Due to the wrapping boundary conditions of the pose cell attractor network, a single pose cell can correspond to multiple physical locations. The experience map resolves this ambiguity by maintaining a graph-based structure where each node represents a distinct experience, defined by a specific combination of pose and visual context [17].

The map is constructed incrementally using the information obtained from the `/topological_action` topic. The action type and the experience IDs are used to determine when a new experience node should be created, when an edge should be added between previously visited locations (loop closure), or when the current node should simply be updated. At the same time, odometry data from the `/odom` topic is integrated to estimate the relative displacement between consecutive experiences, allowing directed links to be established between nodes and maintaining a consistent topological structure of the explored environment. The whole model is described in [15].

G. Visual Odometry Node

The Visual Odometry module is the component responsible for estimating the motion and position of a camera (or robot) over time by analyzing changes in sequentially captured images. It operates by detecting and tracking visual features between consecutive frames, calculating the geometric transformation required to align these features and, consequently, determining the camera's relative displacement.

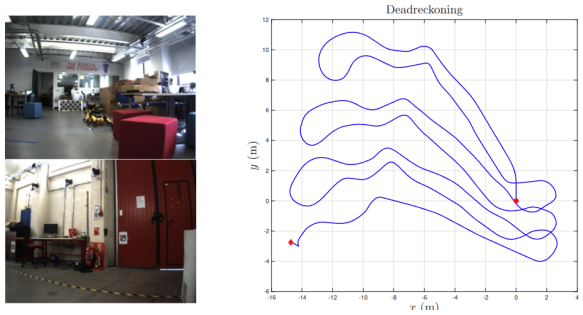

Fig. 2: (a) Frames examples from the frontal camera (b) Odometry from Husky at Robotarium experiment.

The Visual Odometry node subscribes to an `/image_raw` topic and publishes an `/odom` topic. This module is optional and can serve as an alternative to other odometry sources. The whole model is described in [4].

III. Experimental Results

This section presents the results obtained from experiments with four different datasets. Robotarium and iRat Australia are terrestrial datasets, while FIU MMC Lake and FIU BBC Pier are aquatic datasets collected by an USV (Unmanned Surface Vehicle). The results were evaluated using the Euclidean distance between the ground truth and the generated trajectories at each temporal sample.

Both NeoSLAM and RatSLAM rely on a large number of parameters. To support reproducibility and assist in system tuning, this section discusses the role of the most relevant parameters and the expected effects of adjusting them. A complete list of parameters along with their default values is available on the project page [5].

A. Robotarium Dataset

This dataset was created at the Robotarium laboratory at Heriot-Watt University and was introduced in [10]. Although NeoSLAM has already been tested on this dataset, the parameters were not reported. A Husky A200 UGV (Unmanned Ground Vehicle) was used to generate and collect the data. Images were captured using only the left camera (Figure 2-(a)), while odometry readings were obtained from the wheel encoders. A noticeable orientation error in the odometry between laps can be observed in Figure 2-(b), as the trajectories would otherwise overlap. Figure 3 shows that both algorithms were able to generate consistent topological maps of the indoor environment.

Figure 4 compares the number of experiences and view cells created by the algorithms. On the left, the experiences (in blue) and view cells (in red) generated by RatSLAM are shown. On the right, the experiences and spatial-view cells generated by NeoSLAM are presented. It is important to note that the number of view cells created by NeoSLAM was lower than that of RatSLAM, indicating that NeoSLAM was

[5] https://neurosciencerobotics.github.io/neoslam.github.io/

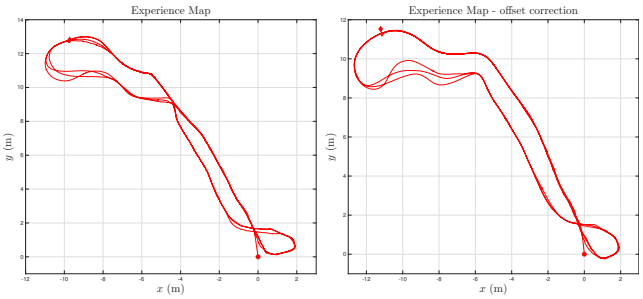

Fig. 3: Experience map for Robotarium dataset. (a) Result of RatSLAM algorithm and (b) NeoSLAM results.

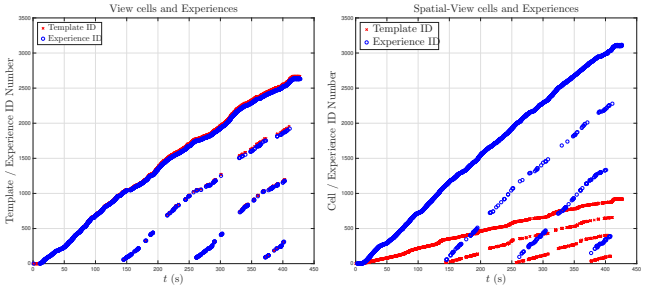

Fig. 4: Visual Cells and experiences for Robotarium. (a) RatSLAM on the left and (b) NeoSLAM on the right

able to build a consistent map with fewer visual comparisons by combining sequential visual scenes within the same intervals.

B. iRat Australia Dataset

The iRat Australia dataset is presented in [16]. The iRat robot has a differential wheel drive, a forward-facing camera, and wheel encoders, which provide odometric readings. A camera mounted overhead captured images that enabled the extraction of ground-truth information. Figure 5 shows an overhead image and frontal camera samples.

Unlike [16], Figure 6-(a) shows the ground truth obtained from the overhead camera and the maximum error between the two trajectories, in addition to the experience map generated by RatSLAM. Figure 6-(b) presents the result obtained with NeoSLAM. For this dataset, the maximum error

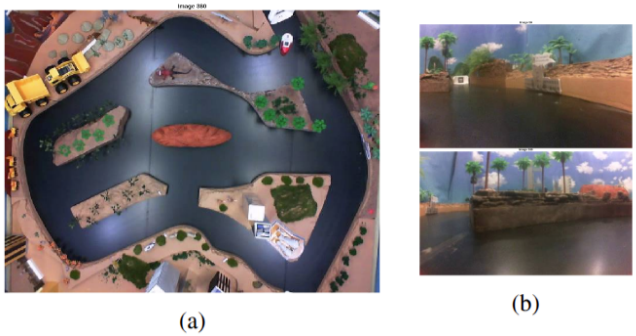

Fig. 5: iRat Australia: (a) Overhead camera and (b) frontal camera frames.

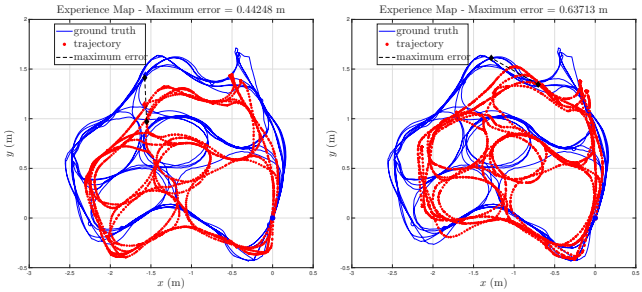

Fig. 6: Experience map for iRat Australia dataset by RatSLAM (a) and NeoSLAM (b).

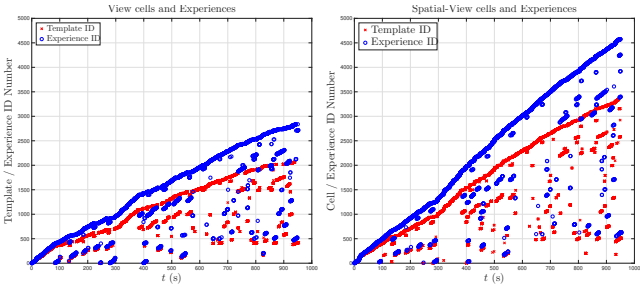

Fig. 7: Visual Cells and Experiences for iRat Australia dataset. (a) RatSLAM on the left and (b) NeoSLAM on the right

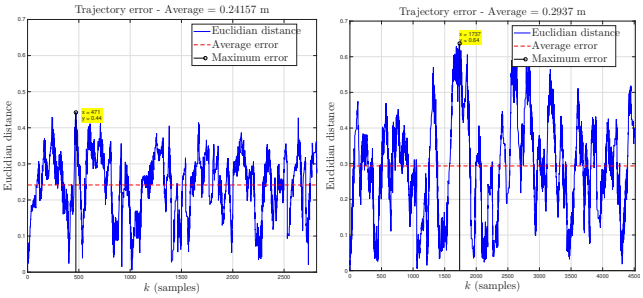

Fig. 8: Trajectory error for iRat Australia (a) RatSLAM and (b) NeoSLAM.

of NeoSLAM was slightly higher than that of RatSLAM. Nevertheless, both maps are very similar and topologically consistent. In contrast to the Robotarium, NeoSLAM generated a greater number of experiences than RatSLAM for the iRat Australia dataset (Figure 7). It is important to note that the number of experiences and templates generated depends on the parameters selected for each dataset. For example, the parameters `theta_alpha` and `theta_rho` influence and control the size of the intervals. Reducing `theta_alpha` and increasing `theta_rho` result in fewer intervals and, consequently, fewer visual cells being created.

Figure 8 compares the trajectory errors for both algorithms. Since we have a reference trajectory for this dataset, the absolute error was calculated for each position between the estimated trajectory and the ground truth, which were temporally synchronized. RatSLAM was slightly superior to NeoSLAM in this dataset according to this metric.

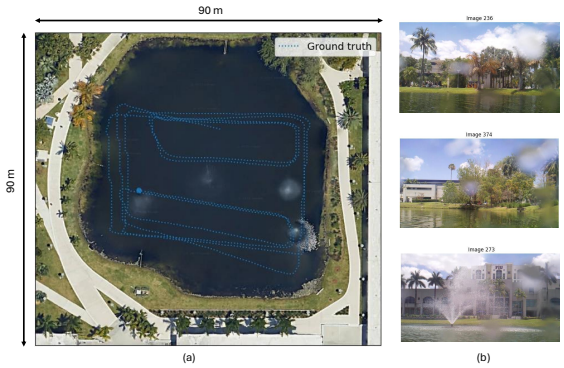

Fig. 9: (a) FIU MMC Lake Dataset and (b) frames examples from the frontal camera.

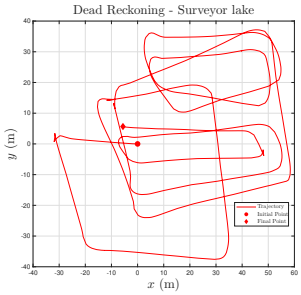

Fig. 10: Odometry from Surveyor ASV at Green Library Lake - Florida International University.

C. FIU MMC Lake Dataset

The acquisition of this dataset was detailed in [15]. Data were collected using a SeaRobotics Surveyor USV at the Green Library Lake at Florida International University (FIU), Modesto Maidique Campus (MMC). Figure 9 shows the trajectory followed by the USV, which covers approximately 900 meters over approximately 16 minutes, along with some sample images from the frontal camera. The dead reckoning can be observed in 10.

Figure 11 shows the trajectories estimated by both algorithms. On the left, we used the same parameters as in [15], and the result was consistent, represented by the red line. However, the maximum error between the experience map and the ground truth is also shown. It can be observed that the result obtained with NeoSLAM was slightly superior to that of RatSLAM. To achieve this outcome, it was necessary to increase the parameter `exclude_recent_intervals`; otherwise, the robot incorrectly relocalized with nearby earlier parts of the trajectory.

A comparison between Figure 12-(a) and (b) reveals that the number of experiences and view cells generated by both algorithms in this experiment was similar, with a slight advantage for NeoSLAM. Analyzing the trajectory errors shown in Figure 13, there was no significant improvement.

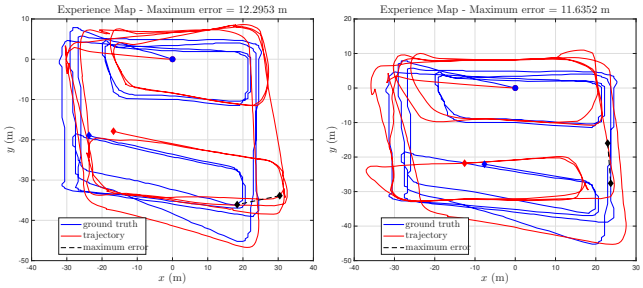

Fig. 11: Experience map of the FIU MMC Lake dataset by RatSLAM (a) and NeoSLAM (b).

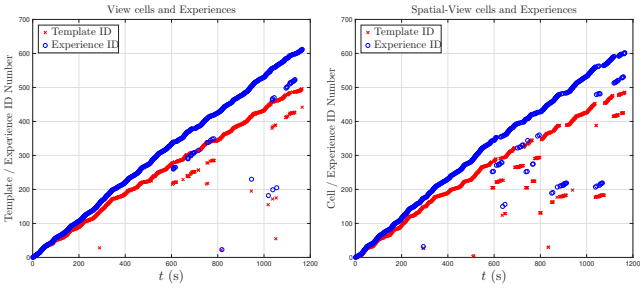

Fig. 12: Cells and Experiences of the FIU MMC Lake Dataset. (a) RatSLAM on the left and (b) NeoSLAM on the right

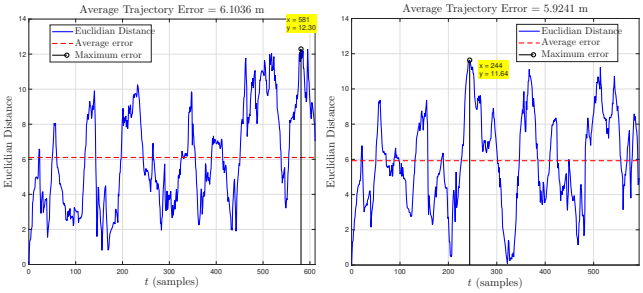

Fig. 13: Trajectory Errors for the FIU MMC Lake dataset using (a) RatSLAM and (b) NeoSLAM.

D. FIU BBC Pier Dataset

This section presents the results from the experiment conducted at the Biscayne Bay Campus (BBC) Pier at FIU. Figure 14 shows the trajectory followed by the USV (Unmanned Surface Vehicle) over approximately 35 minutes, at an average speed of 1 m/s. Figure 15 shows substantial odometry drift for this dataset.

This dataset is more challenging because the environment contains few distinct features. The images are very similar along the trajectory, which increases the likelihood of false positives. After an extensive parameter tuning process, RatSLAM generated a result without map collapse, as shown in Figure 16-(a). In contrast, NeoSLAM was able to produce a trajectory closer to the ground truth. The Maximum trajectory error was significantly lower with NeoSLAM (34.44 m) than with RatSLAM (55.34 m), a reduction of 38 %. Figure 17 shows that the number of experiences created by both algorithms was similar; however, NeoSLAM generated

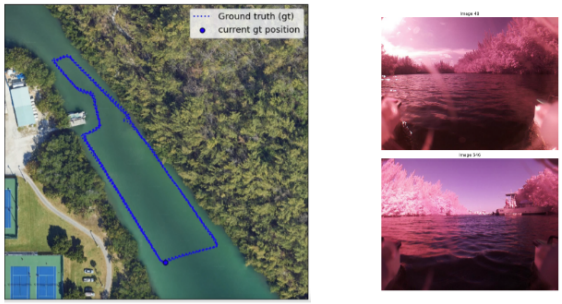

Fig. 14: (a) FIU BBC Pier Dataset and (b) frames examples from the frontal camera.

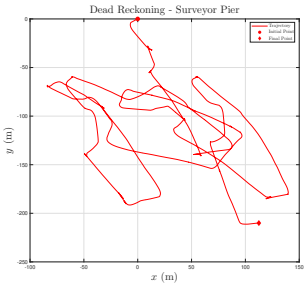

Fig. 15: Odometry from Surveyor ASV at FIU BBC Pier.

fewer spatial-view cells. The results in Figure 18 show that NeoSLAM also achieved better performance, with an Average Error of 15.10 m compared to 18.45 m for RatSLAM.

IV. Conclusion

This paper introduced a new implementation of NeoSLAM that improves computational efficiency and prevents real-time data loss. The proposed modular architecture facilitates scalability, maintainability and integration with modern tools, while new parameters were incorporated to increase robustness. The approach was compared with RatSLAM across four datasets. To the best of our knowledge, this is the first work evaluating NeoSLAM on USV datasets, which are particularly challenging due to the purely inertial odometry source and the significant visual ambiguity. Even under these conditions, both NeoSLAM and RatSLAM were able to construct topological maps that improve upon the degraded pure odometry trajectory. Considering the reconstructed Cartesian trajectories and the number of view cells created, both methods achieved similar performance, with NeoSLAM showing a slight advantage for the datasets evaluated in this paper. For future work, internal stages of the NeoSLAM pipeline will be further investigated to better understand its influence on loop closure detection, such as the similarity among visual features. Additionally, although included in the architecture, the Spatial Pooler component was not used in the present experiments; its incorporation is expected to improve visual input discrimination and increase robustness to noise.

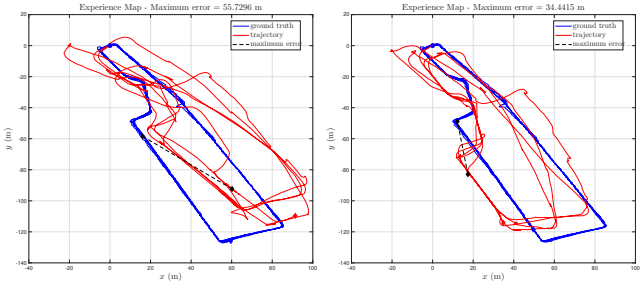

Fig. 16: Experience map for FIU BBC Pier dataset by RatSLAM (a) and NeoSLAM (b).

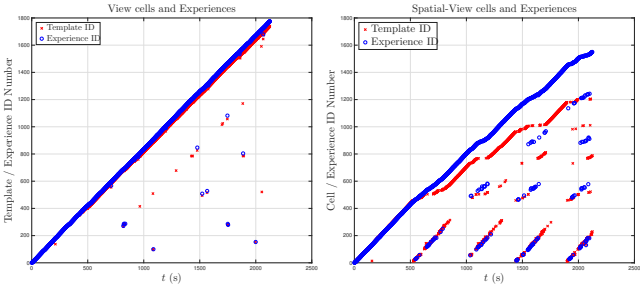

Fig. 17: Visual Cells and Experiences for the FIU BBC Pier Dataset. (a) RatSLAM on the left and (b) NeoSLAM on the right.

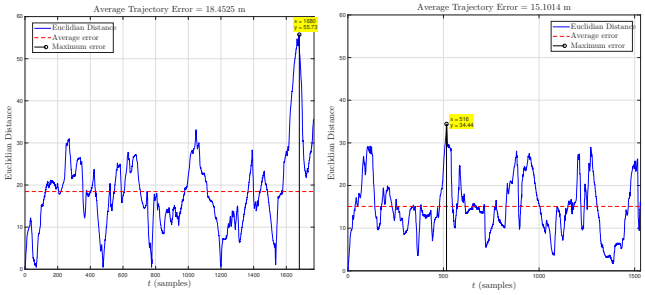

Fig. 18: Trajectory Errors for the FIU BBC Pier dataset using (a) RatSLAM and (b) NeoSLAM.

REFERENCES

[1] K. Baxevani, I. Yadav, Y. Yang, M. Sebok, H. G. Tanner and G. Huang, "Resilient Ground Vehicle Autonomous Navigation in GPS-Denied Environments". Guidance, Navigation and Control, vol. 2, no. 4, art. 2250020, Nov. 23 2022.

[2] S. Thrun and M. Montemerlo, "The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures". International Journal of Robotics Research, vol. 25, no. 5-6, pp. 403–429, 2006. DOI: 10.1177/0278364906065387.

[3] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147–1163, Oct. 2015. DOI: 10.1109/TRO.2015.2463671.

[4] M. J. Milford, G. F. Wyeth and D. Prasser, "RatSLAM: a hippocampal model for simultaneous localization and mapping," IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, New Orleans, LA, USA, 2004, pp. 403-408 Vol.1.

[5] M. J. Milford and G. F. Wyeth, "Mapping a Suburb With a Single Camera Using a Biologically Inspired SLAM System," in IEEE Transactions on Robotics, vol. 24, no. 5, pp. 1038-1053, Oct. 2008.

[6] J. O'Keefe and L. Nadel, "The hippocampus as a cognitive map". Oxford, United Kingdom: Clarendon Press, 1978.

[7] J. R. Ranck Jr., "Head direction cells in the deep cell layer of dorsal presubiculum in freely moving rats". Society for Neuroscience Abstracts, vol. 10, p. 599, 1984.

[8] T. Hafting, M. Fyhn, S. Molden, M.-B. Moser and E. I. Moser, "Microstructure of a spatial map in the entorhinal cortex". Nature, vol. 436, pp. 801–806, 2005. DOI: 10.1038/nature03721.

[9] T. Solstad, E. I. Moser and G. T. Einevoll, "From grid cells to place cells: A mathematical model". Hippocampus, vol. 16, no. 12, pp. 1026–1031, 2006.

[10] C.A.P. Pizzino, R.R. Costa, D. Mitchell and P.A. Vargas, "NeoSLAM: Long-Term SLAM Using Computational Models of the Brain". Sensors, 2024, 24, 1143.

[11] Y. Cui, S. Ahmad and J. Hawkins, "Continuous online sequence learning with an unsupervised neural network model". Neural Computation, vol. 28, no. 11, pp. 2474–2504, 2016.

[12] S. Ahmad and J. Hawkins, "Properties of Sparse Distributed Representations and their application to Hierarchical Temporal Memory". Technical Report, Numenta, 2015.

[13] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, San Francisco, CA, USA, 2001, pp. 245–250.

[14] P. Neubert, S. Schubert and P. Protzel, "A Neurologically Inspired Sequence Processing Model for Mobile Robot Place Recognition," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3200–3207, 2019, doi: 10.1109/LRA.2019.2927096.

[15] F. Coelho, J. V. T. Borges, P. Padrao, J. Fuentes, R. R. Costa, L. Hsu and L. Bobadilla, "Bioinspired SLAM Approach for Unmanned Surface Vehicle". In: Proceedings of the International Conference on Advanced Robotics (ICAR), San Juan, Argentina, Dec. 2–5, 2025 (preprint on arXiv: abs/2509.19522), 2025. arXiv:2509.19522.

[16] D. Ball, S. Heath, J. Wiles, G. Wyeth, P. Corke and M. Milford, "OpenRatSLAM: an open source brain-based SLAM system," Auton Robot 34, 149–176, 2013.

[17] G. Wyeth and M. Milford, "Spatial cognition for robots," *IEEE Robotics & Automation Magazine*, vol. 16, no. 1, pp. 24–32, March 2009.

[18] E. T. Rolls, "A Theory and Model of Scene Representations With Hippocampal Spatial View Cells". Hippocampus, vol. 35, no. 3, art. e70013, 2025. DOI: 10.1002/hipo.70013.

[19] Y. Cui, S. Ahmad, and J. Hawkins, "The HTM Spatial Pooler—A Neocortical Algorithm for Online Sparse Distributed Coding," *Frontiers in Computational Neuroscience*, vol. 11, article 111, 2017, doi:10.3389/fncom.2017.00111.