Bioinspired SLAM Approach for Unmanned Surface Vehicle

Fabio Coelho[1], Joao Victor T. Borges[1], Paulo Padrao[2], Jose Fuentes[3], Ramon R. Costa[1], Liu Hsu[1] and Leonardo Bobadilla[3]

***Abstract*—This paper presents OpenRatSLAM2, a new version of OpenRatSLAM—a bioinspired SLAM framework based on computational models of the rodent hippocampus. OpenRatSLAM2 delivers low-computation-cost visual-inertial based SLAM, suitable for GPS-denied environments. Our contributions include a ROS2-based architecture, experimental results on new waterway datasets, and insights into system parameter tuning. This work represents the first known application of RatSLAM on USVs. The estimated trajectory was compared with ground truth data using the Hausdorff distance. The results show that the algorithm can generate a semimetric map with an error margin acceptable for most robotic applications.**

***Index Terms*—SLAM, RatSLAM, Bioinspired Navigation, USV, ROS 2, Visual-Inertial SLAM**

I. Introduction

The increasing use of unmanned surface vehicles (USVs) for scientific, military, and commercial purposes requires the development of robust navigation systems [1]. Common applications include oceanographic data collection, oil and gas exploration, environmental surveys, mine countermeasures, and surveillance [2], [3]. To autonomously perform such tasks, a mobile robot must be able to localize itself within its environment [4]. Common approaches include combining GPS with an inertial measurement unit (IMU) and Kalman filtering algorithms for state estimation in USVs [5]– [6]. However, these methods fail in GPS-denied environments where satellite signals are obstructed [7]. Moreover, GPS signals are vulnerable to various disruptions and cyberattacks, including jamming and spoofing [8]. To address these limitations, Simultaneous Localization and Mapping (SLAM) is an alternative that enables a vehicle to build a map of its surroundings while estimating its position relative to that map. Many existing SLAM implementations rely on computationally intensive sensors, such as LiDAR or depth cameras. These sensors often require high processing and storage demands, making them less suitable for real-time applications on resource-constrained platforms [9]. Motivated by recent advances in neuroscience, several brain-inspired SLAM systems have been proposed [10]. A pioneering work is the RatSLAM framework, a biologically inspired SLAM algorithm based on computational models of the rodent hippocampus. RatSLAM employs a Continuous Attractor Neural Network (CANN) to construct a cognitive map of an environment using only a low-resolution monocular camera [11]. Compared to probabilistic SLAM approaches, RatSLAM offers reduced computational complexity and efficient memory usage and it is well-suited for both indoor and large-scale outdoor mapping. In recent years, several RatSLAM-based variants have been proposed [12]. For instance, [13] introduced a MATLAB-based RatSLAM implementation in a rat robot, demonstrating its capability to learn spatial layouts. However, the system's performance was too slow for real-time operation in large environments. Another approach, OpenRatSLAM, was proposed as an open-source RatSLAM implementation based on the Robot Operating System (ROS) [14]. This version benefits from ROS's node parallelization and modular integration with diverse robotic architectures [12]. The emergence of ROS 2 as the dominant middleware for new robotic systems has created integration challenges, as OpenRatSLAM was primarily developed for ROS 1. In this context, xRatSLAM was developed as an extensible, parallel, open-source framework developed as a C++ library to facilitate the development and testing of RatSLAM algorithm variants [12]. While most applications targeted ground robots, RatSLAM-inspired algorithms have also been explored in other domains. One aerial application, NeuroSLAM, is a neuro-inspired SLAM system with four degrees of freedom (4DoF), based on computational models of 3D grid cells and multilayered head direction cells. It integrates visual and self-motion cues through a dedicated vision system [15]. In underwater environments, two RatSLAM-based systems have been developed: DolphinSLAM [16], a 3D variant, and a more recent system that implements Pose Cells using Spiking Neural Networks (SNNs) [17]. Both were developed using ROS 1 distributions, which are now deprecated and unsupported. In summary, the contributions of this work are as follows:

- A new version of OpenRatSLAM, implemented using ROS 2 Rolling, referred to as OpenRatSLAM2. This version benefits from ROS 2's advantages, including improved maintainability and easier integration with modern tools. Also, the communication middleware is more robust than the old ROS 1, providing streamlined transition from simulation to physical robot deployment;

[1] Fabio Coelho, Joao Victor T. Borges, Ramon Romankevicius, and Liu Hsu are with the Department of Electrical Engineering, Federal University of Rio de Janeiro, Brazil f.coelho@gsuite.iff.edu.br, {borgesjvt, ramonrcosta}@gmail.com, lhsu@coppe.ufrj.br

[2] Paulo Padrao is with the Department of Mathematics and Computer Science, Providence College, Providence, RI, USA ppadraol@providence.edu

[3] Jose Fuentes and Leonardo Bobadilla are with the School of Computing and Information Sciences, Florida International University, Miami, FL, USA jfuen099@fiu.edu, bobadilla@cs.fiu.edu

- To the best of our knowledge, this is the first application of RatSLAM to an USV;
- A visual-inertial dataset collected using a USV for evaluating SLAM performance in aquatic environments.

II. Problem Formulation

RatSLAM is an appearance-based system introduced in [18] that relies on visual similarity between images captured at discrete locations in the environment. It is a mapping and localization algorithm inspired by the neural processes associated with spatial navigation in the hippocampus and entorhinal cortex of rodents [19]. Figure 1 illustrates the OpenRatSLAM architecture, which builds upon the original RatSLAM framework. The system consists of three main modules: *Local View Cells*, *Pose Cells*, and the *Experience Map*.

The *Local View Cells* are visual templates that represent unique scenes learned in the environment. The *Pose Cells Network* is the core of the algorithm and it models the behavior of three types of cells found in the rodent brain, strongly linked to spatial location: *place*, *head* and *grid* cells.

Place cells fire at their peak when the rodent is in a specific location, with excitation decreasing as the animal moves away from that location [11]. *Head direction cells* activate only when the animal is oriented toward specific global directions [20], [21].

The activity packet in the pose cell network encodes the belief about the current pose, denoted by the vector $\begin{bmatrix} x & y & \theta \end{bmatrix}^T$. Each local view cell is anchored, at the time of its creation, to the centroid of an active place cell packet. This association is indicated by the brown lines in Fig. 1.

The *Experience Map* is a topological graph-based representation that integrates information from pose cells and local view cells. Each node, or experience, stores the estimated robot pose, derived from the centroid of the activation packet, and the corresponding local view ID, both captured at the time of experience creation. These associations are illustrated by the blue lines in Fig. 1.

Over time, odometry-based dead reckoning accumulates drift. For instance, a gray node in the experience map may represent the estimated pose based on odometry alone, while a matching visual template indicates the scene corresponds to a previously observed location (e.g., experience zero). This triggers the loop closure process, which is described in subsection II-C.

A. Pose Cells Network

Pose Cells Network (PCN) is a continuous attractor network (CAN) implemented as a three-dimensional structure of cells with weighted excitatory and inhibitory connections [22]. It exhibits characteristics similar to those of navigation-related neurons found in several mammalian brains, particularly grid cells. Cells within the network are locally linked via excitatory connections that wrap around all six faces of the network. Additionally, each cell inhibits all other cells in the network, which contribute to the formation of localized activity packets.

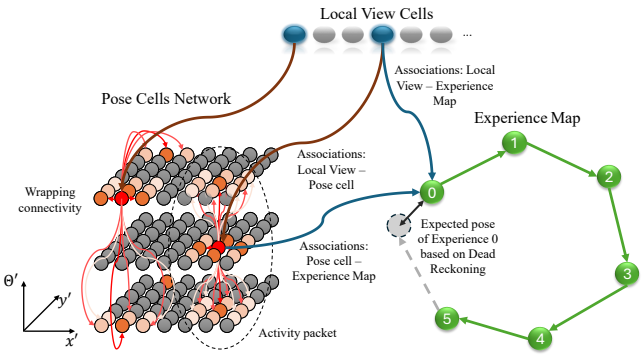

Fig. 1: RatSLAM main modules.

[14]. Unlike most artificial neural networks, the CAN model does not update its state by adjusting connection weights. Instead, it is updated by varying the activity levels of its neural units [22].

1) Attractor Dynamics: The intrinsic attractor dynamics of the network ensure that, in the absence of external input, the activity converges over several iterations to a single localized packet. For each pose cell, *local excitation* is achieved through a three-dimensional Gaussian distribution of weighted connections. This process is visually represented by the red arrows in Fig. 1. In the OpenRatSLAM implementation, the excitatory weight matrix ε is given by Eq. (1)

$$\varepsilon_{a,b,c} = \frac{1}{\sigma_\varepsilon \sqrt{2\pi}} \mathrm{e}^{\left(\frac{-(a-x_c)^2 (b-y_c)^2 (c-\theta_c)^2}{2\sigma_\varepsilon^2} \right)}, \tag{1}$$

where a, b, and c represent the distances between cells in x', y' and θ' coordinates, respectively; σ_ε denotes the variance of the excitation kernel; and $\begin{bmatrix} x_c & y_c & \theta_c \end{bmatrix}^T$ is the center of the activity distribution.

The resulting change in the activity of a pose cell due to local excitation is given by Eq. (2)

$$\Delta P_{x',y',\theta'} = \sum_{i=0}^{n_{x'}-1} \sum_{j=0}^{n_{y'}-1} \sum_{k=0}^{n_{\theta'}-1} P_{i,j,k} \varepsilon_{a,b,c}, \tag{2}$$

where $n_{x'}$, $n_{y'}$ e $n_{\theta'}$ denote the dimensions of the pose cell matrix in units of cells. Equation 2 represents a circular convolution of two three-dimensional matrices.

The following steps are *local* and *global inhibition*. Each cell inhibits nearby cells using an inhibitory kernel $\psi_{a,b,c}$, which is a broader version the excitation kernel. The variances for inhibition are larger than for excitation, creating the so-called Mexican-hat function. Furthermore, a constant global inhibition φ is uniformly applied across all cells. The total inhibitory update is described by Eq. (3)

$$\Delta P_{x',y',\theta'} = - \sum_{i=0}^{n_{x'}-1} \sum_{j=0}^{n_{y'}-1} \sum_{k=0}^{n_{\theta'}-1} P_{i,j,k} \psi_{a,b,c} - \varphi, \tag{3}$$

where φ is the global inhibition constant. After inhibition, all pose cells values P are clipped to be nonnegative and then *normalized* so that the total energy in the network sums to one [11].

2) Path Integration: In the RatSLAM system, path integration consists of shifting the activity packet across the pose cell network based on odometry information. Although this approach presents lower biological fidelity compared to computing transitions through weighted connections, it is computationally efficient and avoids scalability issues. Unlike probabilistic SLAM approaches, this mechanism does not increase uncertainty over time. Odometry data used for path integration can be extracted from image-based motion estimation or obtained from other odometric sources. The accumulated error in the path integration process is reduced by the activation of local view cells, which inject energy into the pose cell network when familiar scenes are detected, thereby enabling loop closure and correction of the robot's estimated pose.

B. Local View Cells

Local view cells consist of an expandable array of units, denoted by V, where each cell encodes a distinct visual scene. A given cell becomes active when the robot perceives its corresponding scene. While there are no direct connections between local view cells themselves, connections are formed between local view cells and pose cells upon creation of a new visual template. When a new local view cell V_i is created, an excitatory link β_i is learned between this unit and the centroid of the currently dominant activity packet in the pose cell network. If the same visual scene is encountered again, the associated local view cell injects activity into the pose cells through this excitatory link. This process is described by Eq. (4)

$$\Delta P_{x',y',\theta'} = \delta \sum_i \beta_{i,x',y',\theta'} V_i. \tag{4}$$

where δ is a constant that determines the influence of visual cues on the correction of robot's pose estimate. A saturation mechanism is implemented to limit the duration for which a visual template can inject activity, preventing erroneous relocations in the absence of movement. Successful relocalization requires the robot to perceive a familiar sequence of images, resulting in a series of energy injections into the same region of the pose cell network. The visual energy injection process is critical, and its performance is sensitive to parameter tuning as provided in Section IV.

C. Experience Map

Although pose cells represent a finite area, the wrapping of the network edges allows an infinite area to be mapped. As a result, a single pose cell may correspond to multiple physical locations. To solve potential ambiguities, the experience map is a semi-metric topological map [22], that estimates a unique pose of the robot by combining information from pose cells, local view cells and odometry. The experience map consists of a graph, where each node (referred to as *experience*) is defined as a 3-tuple $e_i = \left\{P^i, V^i, \mathbf{p}^i\right\}$, where P^i and V^i are the pose cell and local view activity states, respectively, at the time of experience creation. The term $\mathbf{p}^i$ represents the estimated pose of the robot within the coordinate space of the experience map.

1) Experience Creation: The robot's current pose and local view information are compared against all previously stored experiences through a matching score metric S^i provided by Eq. (5)

$$S^i = \mu_p \left|P^i - P\right| + \mu_v \left|V^i - V\right|. \tag{5}$$

μ_p and μ_v are weighting factors for the pose and local view components, respectively. If the minimum score across all stored experiences satisfies $\min(\mathbf{S}) \geq S_{\text{max}}$, indicating that the current state is sufficiently distinct, a new experience is created.When a new experience e_j is added to the graph, a transition l_{ij} is also established between e_j and the previously active experience e_i. This transition is represented in Eq. (6)

$$l_{ij} = \left\{\Delta \mathbf{p}^{ij}, \Delta t^{ij}\right\}, \tag{6}$$

where $\Delta \mathbf{p}^{ij}$ denotes the relative pose change computed from odometry, and Δt^{ij} is the elapsed time since the last experience. The new experience e_j is given by Eq. (7)

$$e_j = \left\{P^j, V^j, \mathbf{p}^i + \Delta p^{ij}\right\}. \tag{7}$$

Equation (7) T is valid only at the time of experience creation. The value of $\mathbf{p}^j$ may subsequently be adjusted during loop closure.

2) Loop Closure: If any of the stored experience matching scores fall below the threshold S_{max}, the experience with the lowest score is chosen as the active experience. This experience represents the best estimate of the robot's current location, thereby triggering the loop closure process. At this point, the relative pose between the two matched experiences typically differs from the pose change predicted by odometry. In this case, an *experience map relaxation* procedure is applied. This process minimizes the error between the observed transitions and the absolute poses of experiences within the map. The pose of all experiences are updated using Eq. (8)

$$\Delta \mathbf{p}^i = \alpha \left[\sum_{j=1}^{N_f} \left(\mathbf{p}^j - \mathbf{p}^i - \Delta \mathbf{p}^{ij}\right) + \sum_{k=1}^{N_t} \left(\mathbf{p}^k - \mathbf{p}^i - \Delta \mathbf{p}^{ki}\right)\right], \tag{8}$$

where α is a correction rate constant, N_f is the number of links from experience e_i to others, and N_t is the number of links from other experiences to e_i. A value of $\alpha = 0.5$ ensures a balance between the velocity of the correction and the stability of the map, while higher values of α may cause instability [23].

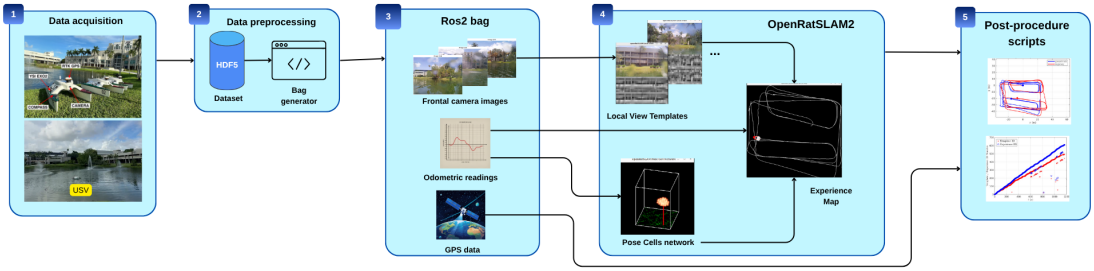

Fig. 2: OpenRatSLAM2 workflow: (*i*) Data acquisition while the USV travels a predetermined trajectory; (*ii*) Data preprocessing consists of adapting the input format to the OpenRatSLAM2 framework; (*iii*) A ROS2 bag file containing the data from the sensors of interest; (*iv*) RatSLAM main modules, where input data are processed to generate the experience map; (*v*) Scripts for viewing and analyzing results.

III. System Overview

In this work, we employed the same data formats used in the first version of OpenRatSLAM. Input data, such as odometric readings and camera images, are read from a ROS bag. Figure 2 shows the general workflow. Data collection was performed by the SeaRobotics Surveyor (Fig. 2-(i)), and initially stored in Hierarchical Data Format version (HDF5). Next, the HDF5 data file was converted to the ROS 2 bag format to be processed by the OpenRatSLAM2 framework. During OpenRatSLAM2 running, the topic data are recorded into a separate ROS bag file. Output data, such as the generated map, are extracted and visualized using a set of post-processing scripts.[1]

Figure 2 shows the three sensors used to compose the datasets: compass, camera, and GPS. The ISA500 compass includes an integrated Attitude and Heading Reference System (AHRS), with three MEMS-based gyroscopes, accelerometers, and magnetometers and provides the odometric readings. The onboard camera captures frontal-view images, while the GPS is used to generate the *ground truth*.

OpenRatSLAM2 follows the same architectural design as OpenRatSLAM [14], which consists of four ROS 2 nodes, as depicted in Fig. 3:

- *Visual Odometry*: provides an odometry estimate based on image changes. This node not used in this work, as the dataset includes an alternative odometric source;
- *Local View Cells*: verifies whether the current view corresponds to a previously encountered scene;
- *Pose Cell Network*: the core node of the system; manages the activity packet to estimate pose based on odometric and local view connections. It also handles the experience map nodes and links creation.
- *Experience Map*: builds experience graph, performs graph relaxation, and handles path planning.

IV. OpenRatSLAM2 Parameters

OpenRatSLAM's performance depends on a large set of parameters. To support reproducibility and facilitate system

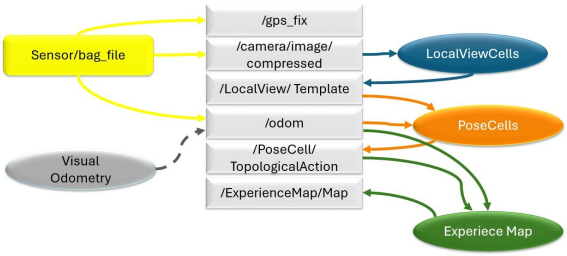

Fig. 3: ROS Computational Graph.

tuning, this section presents the influence of the main parameters and the expected impact of modifying them. All parameters and their default values can be seen on the project page[2].

Regarding the Local View module's parameters, *image_crop* parameters define regions of the image relevant for localization. In this dataset, for example, the water surface can be excluded. The *template_size* parameters define the spatial resolution of the template. Small values may fail to capture relevant features, while large values increase memory and computation and may introduce over-sensitivity.

Normalization parameters mitigate the effects of lighting variation. For example, *vt_normalization* ≈ 1 maintains the original contrast, adjusting only the brightness. High values (> 2), on the other hand, can cause saturation.

Among the comparison parameters, *vt_match_threshold* is critical: lower values (0.01–0.03) reduce false positives and increase the number of templates created—suitable for repetitive environments. Higher values (0.05-0.1) tolerate greater variation, but risk confusing different locations.

Most pose cell parameters do not require frequent tuning. *pc_dim_x* sets the spatial resolution of the pose cell network. A larger network improves distinctiveness and loop closure accuracy at the cost of increased computation. *pc_cell_x_size* should match the robot's speed profile—ideally, the energy packet moves one cell per iteration.

For static environments or slow-moving robots,

[1] https://github.com/BorgesJVT/ratslam/blob/develop/scripts/README.md

[2] INSERTGITHUBLINKHERE

TABLE I: Parameter Values.

Local View	***Pose Cells***
image_crop_x_min = 40	pc_dim_xy = 18
image_crop_x_max = 600	pc_cell_x_size = 1
image_crop_y_max = 150	pc_vt_inject_energy = 0.2
image_crop_y_min = 300	exp_delta_pc_threshold = 2.0
template_x_size = 60	vt_active_decay = 1.0
template_y_size = 20	pc_vt_restore = 0.05
vt_shift_match = 25	
vt_step_match = 5	***Experience Map***
vt_match_threshold = 0.073	exp_loops = 50
vt_normalisation = 0	exp_initial_em_deg = 180
vt_patch_normalise = 2	exp_correction = 0.5
vt_panoramic = 0	

vt_active_decay can be increased to 1.5. In dynamic environments or at higher speeds, a value between 0.3 and 0.8 is recommended. The parameter *pc_vt_inject_energy* regulates how strongly visual input corrects pose estimates—0.1–0.2 is typical for stable conditions, while 0.05–0.1 may be better suited for dynamic scenarios. The *exp_delta_pc_threshold* affects the density of the topological map. Lower values (< 1.5) generate denser, more detailed maps at the cost of memory. Higher values (> 3.0) reduce node creation and computational load, but may reduce environmental resolution. The parameter *exp_loops* defines how many iterations of the relaxation algorithm are executed per system update.

V. Experimental Results

This section presents the results obtained from the experiment conducted using the Green Library Lake dataset, collected at Florida International University (FIU). The dataset includes LiDAR data, odometry, frontal camera images, GPS, and water quality data, all sampled at 1 Hz. The images and odometry data were re-encoded as `CompressedImage` and `Odometry` messages, respectively, and stored in a ROS 2 bag file. During preprocessing, images were resized to 640x480 pixels. To ensure accurate synchronization across data streams, the original timestamps were preserved. Table I summarizes the parameter values used in the experiment.

Figure 4 (a) shows the trajectory followed by the USV, which consists of approximately 900 meters over a duration of about 16 min. Figure 4 (b) presents sample frames captured by the onboard camera. (Figs. 5a) shows the evolution of the experience map over time. At first, the USV had already completed two full outer loops and one inner loop in the upper region of the lake. At this stage, no loop closure had yet occurred, and odometry drift is visible. In the second figure, a loop closure with the starting region is detected, triggering map convergence toward the ground truth. With subsequent loop closures, the map becomes increasingly stable, requiring only minor refinements.

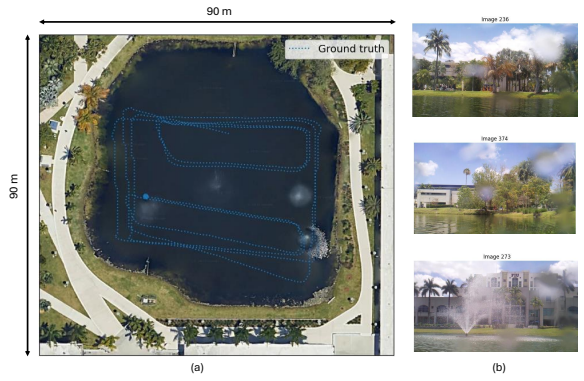

Fig. 4: (a) FIU MMC Lake Dataset and (b) frames examples from frontal camera.

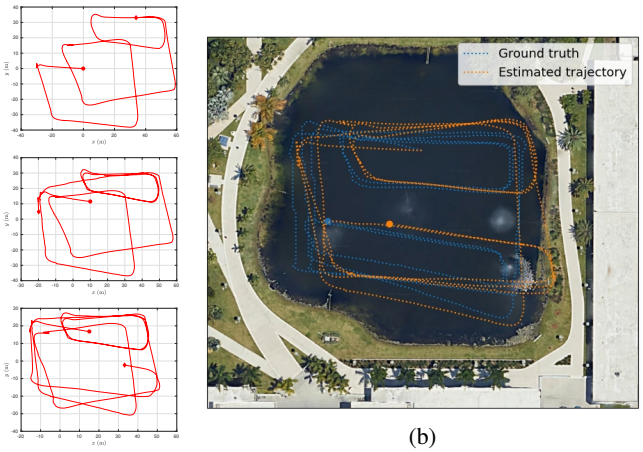

Fig. 5: (a) Experience map evolution over time and (b) final map.

The final map is shown in Figure 5b where the estimated trajectory (orange) closely aligns with the ground truth (blue). A complete visualization of the trajectory evolution is available in the accompanying video.[3] . The accuracy of the estimated trajectory was evaluated using the *Hausdorff distance*. Given a metric space (X, d) and two non-empty subsets $A, B \subseteq X$, the Hausdorff distance $d_H(A, B)$ is defined as Eq. (9):

$$d_H(A,B) = \max\left(\sup_{a\in A}\inf_{b\in B} d(a,b), \sup_{b\in B}\inf_{a\in A} d(a,b)\right). \quad (9)$$

This metric computes the maximum distance from any point in one set to the nearest point in the other set [24]. The Hausdorff distance is particularly suitable for this case when compared to more traditional metrics such as Mean Absolute Error (MAE) or Mean Squared Error (MSE), especially because the estimated trajectory $\boldsymbol{p'}$ and the ground truth $\boldsymbol{g}$ do not necessarily contain the same number of points. This mismatch arises because the number of experience nodes is automatically determined by the algorithm.

For the evaluated 900-meter trajectory, the Hausdorff distance was $d_H(g, p') \approx 8.35$. This is a reasonable value if compared to the accuracy of the onboard GPS receiver (specifically, the Garmin 19X HVS) which offers an accuracy

[3] INSERTGITHUBLINKHERE

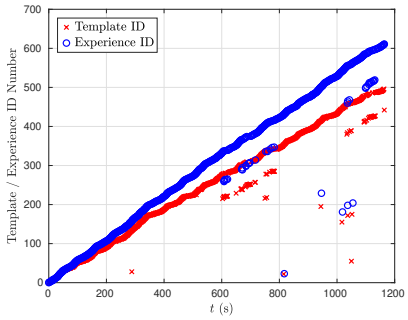

Fig. 6: Active experience and visual template - MMC Lake

of 5–10 meters using GPS alone, and up to 3 meters with Wide Area Augmentation System (WAAS). Figure 6 illustrates the activation timeline of experiences and visual templates throughout the experiment. The upper blue line (bounding line) indicates the creation of new experience nodes, while the red line shows the creation of visual templates. Short segments below these lines indicate successful re-recognition of previously visited locations. Loop closures are marked by the start and end points of these segments.

VI. Conclusion

This work introduced OpenRatSLAM2, a new ROS2-based version of OpenRatSLAM, designed as a modular framework for online and offline operation. Unlike navigation systems that rely on probabilistic methods, laser range sensors, or occupancy maps, RatSLAM works with only a low-resolution monocular camera. It also supports integration of odometric data from additional sensors, such as encoders and IMUs. To the best of our knowledge, this is the first work detailing the application of RatSLAM to an unmanned surface vehicle. Field experiments demonstrated that this approach can localize an USV in GPS-denied scenarios with an error margin acceptable for robotic applications.Future work will explore autonomous or supervised optimization of OpenRatSLAM2's local view and pose cell parameters, using predefined dataset segments with known loop closures.

Acknowledgment

This research was supported in part by NSF grants IIS-2024733 and IIS-2331908, the Office of Naval Research grant N00014-23-1-2789, the U.S. Department of Defense grant 78170-RT-REP, and the Florida Department of Environmental Protection grant INV31.

References

[1] G. Xia and G. Wang, "INS/GNSS Tightly-Coupled Integration Using Quaternion-Based AUPF for USV". Sensors, 2016, 16, 1215.

[2] A. A. R. Newaz, P. Padrao, J. Fuentes, T. Alam, G. Govindarajan, and L. Bobadilla, "LCD-RIG: Limited Communication Decentralized Robotic Information Gathering Systems," *IEEE Robotics and Automation Letters*, vol. 9, no. 11, pp. 10034–10041, 2024,

[3] G. Puthumanaillam, P. Padrao, J. Fuentes, P. Thangeda, W. E. Schafer, J. H. Song, K. Jagdale, L. Bobadilla, and M. Ornik, "TRACE: A Self-Improving Framework for Robot Behavior Forecasting with Vision-Language Models," *arXiv preprint arXiv:2503.00761*, 2025. [Online].

[4] M.C. Menezes , M.E.S. Munoz, E.P. Freitas , S. Cheng, T. Walther, A.A. Neto, P.R.A. Ribeiro and A.C.M. Oliveira, "Automatic Tuning of RatSLAM's Parameters by Irace and Iterative Closest Point," (2020) IECON Proceedings (Industrial Electronics Conference), 2020, art. no. 9254718, pp. 562 - 568.

[5] W. Liu, Y. Liu and R. Bucknall, "A Robust Localization Method for Unmanned Surface Vehicle (USV) Navigation Using Fuzzy Adaptive Kalman Filtering," in IEEE Access, vol. 7, pp. 46071-46083, 2019.

[6] C. Hide, M. Terry and S. Martin, "Adaptive Kalman filtering for low-cost INS/GPS". The Journal of Navigation, 2003. 56. 143 - 152. 10.1017/S0373463302002151.

[7] X. Liu, Z. Hu, Z. Sun, J. Lu, W. Xie and W. Zhang, "A VIO-Based Localization Approach in GPS-denied Environments for an Unmanned Surface Vehicle," 2023 International Conference on Advanced Robotics and Mechatronics (ICARM), Sanya, China, 2023, pp. 912-917.

[8] J. Burbank, T. Greene and N. Kaabouch, "Detecting and Mitigating Attacks on GPS Devices," Sensors, vol. 24, no. 17, art. no. 5529, 2024.

[9] J. Xu, N. Yan and F. Tang, "An Improvement of Loop Closure Detection Based on BoW for RatSLAM," 2022 37th Youth Academic Annual Conference of Chinese Association of Automation (YAC), Beijing, China, 2022, pp. 634-639.

[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] 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.

[12] M.E. Muñoz, M. C. Menezes, E. Freitas, S. Cheng, P.R. Ribeiro, A. Neto and A.C. Oliveira, "xRatSLAM: An Extensible RatSLAM Computational Framework," Sensors 2022, 22, 8305.

[13] D. Ball, S. Heath, M. Milford, G. Wyeth and J Wiles, "A navigating rat animat", Artificial Life XII: Proceedings of the 12th International Conference on the Synthesis and Simulation of Living Systems, ALIFE 2010, pp. 804 - 811.

[14] 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.

[15] F. Yu, J. Shang, Y. Hu and M. Milford. "NeuroSLAM: a brain-inspired SLAM system for 3D environments", Biol Cybern, 2019, Volume 113, pages 515–545.

[16] L. Silveira, F. Guth, P. Drews-Jr, P. Ballester, M. Machado, F. Codevilla, N. Duarte-Filho, S. Botelho, "An Open-source Bio-inspired Solution to Underwater SLAM", IFAC 2015, Pages 212-217.

[17] B. Manuel Pirozzo, M. De Paula, S. Aldo Villar and G. Gabriel Acosta, "Underwater Rat-SLAM with Memristive Spiking Neural Networks," OCEANS 2024 - Halifax, Halifax, NS, Canada, 2024, pp. 1-8.

[18] 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.

[19] M. C. Menezes et al., "Automatic Tuning of RatSLAM's Parameters by Irace and Iterative Closest Point," IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society, Singapore, 2020, pp. 562-568.

[20] J. S. Taube, R. U. Muller and J. B. Ranck Jr, "Head-direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis", 1990, The Journal of neuroscience : the official journal of the Society for Neuroscience. 10. 420-35.

[21] J. S. Taube, R. U. Muller and J. B. Ranck Jr, "Head-direction cells recorded from the postsubiculum in freely moving rats. II. Effects of environmental manipulations", 1990, The Journal of neuroscience : the official journal of the Society for Neuroscience. 10(2). 436-47.

[22] M. J. Milford and G. F. Wyeth, "Persistent Navigation and Mapping using a Biologically Inspired SLAM System", The International Journal of Robotics Research, 2009, 29(9), pp. 1131-1153.

[23] M. Milford, G. Wyeth and D. Prasser, "RatSLAM on the Edge: Revealing a Coherent Representation from an Overloaded Rat Brain", 2006 ,IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 2006, pp. 4060-4065.

[24] B. Sendov, "Hausdorff Distance. In: Beer, G. (eds) Hausdorff Approximations.", 1990, Mathematics and Its Applications', vol 50, Springer, Dordrecht.