POSEIDONSLAM: UNDERWATER SLAM INSPIRED BY A COMPUTATIONAL MODEL OF THE HUMAN BRAIN

Plínio Bernardo dos Santos Melo

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Ramon Romankevicius Costa

Rio de Janeiro
Março de 2026

POSEIDONSLAM: UNDERWATER SLAM INSPIRED BY A COMPUTATIONAL MODEL OF THE HUMAN BRAIN

Plínio Bernardo dos Santos Melo

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientador: Ramon Romankevicius Costa

Aprovada por: Prof. Ramon Romankevicius Costa
Prof. Fernando Cesar Lizarralde
Prof. Silvia Silva da Costa Botelho

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2026

dos Santos Melo, Plínio Bernardo

PoseidonSLAM: Underwater SLAM inspired by a Computational Model of the Human Brain/Plínio Bernardo dos Santos Melo. – Rio de Janeiro: UFRJ/COPPE, 2026.

XXII, 130 p.: il.; 29, 7cm.

Orientador: Ramon Romankevicius Costa

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2026.

Referências Bibliográficas: p. 109 – 115.

1. Neurorobotics. 2. Marine Robotics. 3. SLAM. 4. Sensor Fusion. 5. Autonomous Underwater Vehicles . I. Romankevicius Costa, Ramon. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

À minha esposa Kássia e às minhas filhas Klara e Katarina.

"Crê em ti mesmo, age e verás os resultados. Quando te esforças, a vida também se esforça para te ajudar."

— Francisco Cândido Xavier

Agradecimentos

Gostaria de agradecer primeiramente a Deus, Pai e Criador, pela oportunidade da evolução intelecto-moral frente a esse desafio de Mestrado Acadêmico em Controle, Automação e Robótica - Análise e Projeto de Sistemas de Controle Avançado -, junto ao Programa de Engenharia Elétrica da COPPE/UFRJ.

À minha esposa Kássia e às minhas filhas Klara e Katarina, pelo suporte nos momentos difíceis, e principalmente por alimentarem com amor a chama que me move nessa jornada.

Ao meu orientador acadêmico, Prof. Dr. Ramon R. Costa, que atenciosamente me acolheu como seu pupilo e iluminou com o seu esclarecimento o caminho para o sucesso da realização desse trabalho.

Ao meu orientador técnico, CF(T) Rui Rodrigues de Mello Júnior, que ombro a ombro me ajudou a superar todos os desafios ao longo desse processo.

Aos amigos do Grupo de Simulação e Controle em Automação e Robótica (GSCAR) do Laboratório de Controle e Automação, Engenharia de Aplicação e Desenvolvimento do GSCAR (LEAD), em especial ao Dr. Carlos Alexandre P. Pizzino, por motivar a exploração do meu projeto de pesquisa a partir do desenvolvimento consolidado da sua tese sobre um método biologicamente inspirado no Modelo Computacional do Neocórtex para solução de *long-term Visual SLAM* em ambiente terrestre, o NeoSLAM.

Aos amigos do Laboratório de Sistemas Marítimos Não Tripulados (LabSMNT) do Instituto de Pesquisas da Marinha (IPqM), em especial ao 1ºTen(EN-RM2) João Victor Torres Borges, por compartilhar da sua experiência com Navegação Autônoma em Robótica Móvel.

Aos demais familiares e amigos que me incentivaram e contribuiram para o sucesso dessa missão.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

POSEIDONSLAM: UNDERWATER SLAM INSPIRED BY A COMPUTATIONAL MODEL OF THE HUMAN BRAIN

Plínio Bernardo dos Santos Melo

Março/2026

Orientador: Ramon Romankevicius Costa

Programa: Engenharia Elétrica

A operação autônoma em ambientes subaquáticos confinados e sem GPS impõe desafios críticos de robustez perceptual (*perceptual aliasing*) e associação de dados (*data association*), tornando o *Simultaneous Localization and Mapping* (SLAM) um subsistema central para navegação segura. Fatores como turbidez da água, iluminação variável, baixa textura e recorrência estrutural do ambiente submarino intensificam o *perceptual aliasing*, levando pipelines geométricos à perda de rastreamento e podendo degradar a consistência global do mapa, inclusive em abordagens bio-inspiradas canônicas quando submetidas a associações espúrias.

Esta dissertação propõe o **PoseidonSLAM**, um framework neuro-inspirado para *Visual-Inertial-Acoustic Underwater SLAM* com ênfase em robustez perceptual e estabilidade topológica. O método combina reconhecimento de lugar — *Visual Place Recognition* (VPR) — por verificação temporal de sequências (*sequence-aware VPR*) com um mecanismo de *VPR Guard* que regula a emissão de eventos no Mapa de Experiências, reduzindo a probabilidade de atualizações motivadas por evidência perceptual instável e, consequentemente, mitigando corrupção estrutural do grafo sob condições degradadas.

A arquitetura é composta por dois subsistemas acoplados: (i) um **front-end neocortical**, derivado do *NeoSLAM*, que codifica observações multimodais de câmera e sonar em *Sparse Distributed Representations* (SDR) e emprega *Hierarchical Temporal Memory* (HTM) para produzir identidades de lugar estáveis; e (ii) um **back-end hipocampal**, derivado do *DolphinSLAM*, que integra essas identidades ao *dead-reckoning* para construir e manter um grafo topológico tridimensional consistente do ambiente submarino explorado.

A validação experimental em *software-in-the-loop* foi conduzida em dois eixos. Primeiro, a *Underwater Cave Transformed Database* foi construída como um teste controlado de robustez do VPR sob perturbações programáticas de aparência e ponto de vista, na qual a rede neural convolucional — *Convolutional Neural Network* (CNN) — ShuffleNetV2 x1.0 apresentou melhor desempenho na curva Precision–Recall do que a CNN AlexNet-conv3. Segundo, o PoseidonSLAM foi avaliado no benchmark *Underwater Cave Sonar and Vision Data Set*, evidenciando: (a) rastreabilidade determinística evento–evidência; (b) redução da densidade de eventos e do tamanho final do grafo com *VPR Guard ON* em relação ao *OFF*; e (c) indícios de fechamento de ciclo (*loop closure*) consistentes com o protocolo de *ground truth* adotado.

Em síntese, os resultados indicam que, ao combinar verificação temporal de sequências e regulação da emissão de eventos, o PoseidonSLAM operacionaliza um mecanismo de robustez perceptual capaz de preservar a coerência topológica do mapa e, também, de reduzir o risco de colapso estrutural do grafo em missões subaquáticas sob condições ambientais severas.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

POSEIDONSLAM: UNDERWATER SLAM INSPIRED BY A COMPUTATIONAL MODEL OF THE HUMAN BRAIN

Plínio Bernardo dos Santos Melo

March/2026

Advisor: Ramon Romankevicius Costa

Department: Electrical Engineering

Autonomous operation in confined underwater environments without GPS poses critical challenges in perceptual aliasing and data association, making *Simultaneous Localization and Mapping* (SLAM) a central subsystem for reliable navigation. Factors such as water turbidity, variable illumination, low texture, and structural recurrence of the underwater environment intensify perceptual aliasing, causing geometric SLAM pipelines to lose tracking and potentially degrading global map consistency, including in canonical bio-inspired approaches under spurious associations.

This dissertation proposes **PoseidonSLAM**, a neuro-inspired framework for *Visual-Inertial-Acoustic Underwater SLAM* with emphasis on perceptual robustness and topological stability. The method combines sequence-aware visual place recognition (VPR) by temporal verification of place hypotheses with a *VPR Guard* mechanism that regulates experience-event emission, reducing updates driven by unstable perceptual evidence and mitigating structural corruption of the experience graph under degraded sensing conditions.

The architecture comprises two coupled subsystems: (i) a **neocortical front-end**, derived from *NeoSLAM*, which encodes multimodal camera and sonar observations into *Sparse Distributed Representations* (SDR) and employs *Hierarchical Temporal Memory* (HTM) to produce stable place identities; and (ii) a **hippocampal back-end**, derived from *DolphinSLAM*, which integrates these identities with dead-reckoning to build and maintain a consistent three-dimensional topological graph of the explored underwater environment.

The software-in-the-loop experimental validation followed two axes. First, the *Underwater Cave Transformed Database* was built as a controlled stress test for VPR robustness under programmatic appearance and viewpoint perturbations, in

which the *Convolutional Neural Network* (CNN) ShuffleNetV2 x1.0 demonstrated better performance in the Precision-Recall curve than CNN AlexNet-conv3. Second, PoseidonSLAM was evaluated on the benchmark *Underwater Cave Sonar and Vision Data Set*, showing: (a) deterministic event–evidence traceability; (b) reduced event density and smaller final graph size with *VPR Guard ON* compared to *OFF*; and (c) loop closure indications consistent with the adopted ground truth protocol.

Overall, the results suggest that PoseidonSLAM operationalizes perceptual robustness by combining temporal sequence verification with event-emission regulation, preserving topological map coherence and reducing the risk of structural graph collapse in severe underwater conditions.

Table of Contents

List of Figures

List of Tables

List of Symbols

t : Discrete time index.

i, j, k, q : Indices (e.g., lattice indices for pose cells).

m : Modality index.

$\mathcal{M}_t$: Set of available modalities at time t.

$\mathbb{B}$: Binary set $\{0, 1\}$.

$\mathcal{L}$: Set of instantiated LVC identities.

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$: 3D topological experience graph (nodes and edges).

$\mathbf{o}_t^{(m)}$: Raw observation of modality m at time t.

$f_m(\cdot)$: Feature extractor for modality m (CNN backbone).

$\mathbf{y}_t^{(m)} \in \mathbb{R}^{d_m}$: Dense embedding of modality m.

d_m : Embedding dimensionality for modality m.

L : SDR length (number of bits).

$p = L/2$: Half-length used in Top/Bottom split.

$\mathbf{P}_m$: Fixed LSBH projection matrix for modality m.

$\mathbf{v}_t^{(m)} \in \mathbb{R}^{L/2}$: Projection responses $\mathbf{P}_m^\top \mathbf{y}_t^{(m)}$.

$s \in (0, 1)$: LSBH selection ratio.

$k = \lfloor s(L/2) \rfloor$: Number of selected indices per half.

$\mathbf{s}_t^{(m)} \in \mathbb{B}^L$: Modality SDR (binary sparse code).

$\bigvee$: Bitwise OR (union) operator.

$\mathbf{s}_t^{\text{fused}}$: Fused SDR $\bigvee_{m \in \mathcal{M}_t} \mathbf{s}_t^{(m)}$.

$\mathcal{A}_t$: Active-bit set of the TM-derived code.

$\mathcal{D}_t$: Descriptor set used for LVC matching.

K_t : Current interval/run length for temporal consolidation.

$K_{\max}$: Maximum interval length.

$I(\mathcal{D}_t, \mathcal{D}_\ell)$: Intersection cardinality $|\mathcal{D}_t \cap \mathcal{D}_\ell|$.

$R(\mathcal{D}_t, \mathcal{D}_\ell)$: Normalized overlap $|\mathcal{D}_t \cap \mathcal{D}_\ell|/|\mathcal{D}_t|$.

$I_{\min}$: Minimum intersection (evidence floor).

τ_R : Overlap acceptance threshold.

ℓ : LVC identity.

$\ell^\star$: Best-matching identity.

ℓ_t : Most-active LVC identity at time t.

w : VPR Guard window size.

r : Minimum repetition threshold within the window.

$C_t(\ell_t)$: Repetition count of ℓ_t in last w emissions.

g_t^{VPR} : Gate decision $\mathbb{I}\{C_t(\ell_t) \geq r\}$.

q_t : Consecutive rejection run length.

$L_{\max}$: Leak threshold (forces acceptance after $q_t \geq L_{\max}$).

$\tilde{g}_t^{\mathrm{VPR}}$: Leaked gate.

$\mathbf{p}_t^{\mathrm{odom}} \in \mathbb{R}^3$: Odometry translation at time t.

$\Delta\mathbf{p}_t$: Odometry increment $\mathbf{p}_t^{\mathrm{odom}} - \mathbf{p}_{t-1}^{\mathrm{odom}}$.

ψ : Yaw (heading).

(x, y, z, ψ) : 4D pose coordinates used by pose cells.

$P_{i,j,k,q}$: Pose cell activity at lattice indices (i, j, k, q).

σ_p, σ_ψ : Excitatory Gaussian variances (translation and yaw).

ϕ : Global inhibition level.

$\beta_{\ell,i,j,k,q}$: Learned LVC→pose-cell coupling (Hebbian).

λ : Learning rate for β update.

δ : Injection gain.

$\mathbf{p}_i \in \mathbb{R}^3$: 3D embedding position of node i.

$\Delta\mathbf{p}_{ij}$: Relative displacement on edge (i, j).

w_{ij} : Edge confidence weight.

α : Relaxation step size.

$\mathcal{N}_f(i), \mathcal{N}_t(i)$: Forward/backward neighbors of node i.

$T_i = [R_i, p_i] \in SE(3)$: Body pose at time i.

$R_i \in SO(3)$: Rotation matrix.

$p_i \in \mathbb{R}^3$: Translation (position).

v_i : Velocity.

b_i^g, b_i^a : Gyroscope and accelerometer biases.

$S_i = \{T_i, v_i, b_i^g, b_i^a\}$: ORB-SLAM3 state vector.

ν : SDR vector notation used in the SDR definition section.

b_i : Binary component of ν.

n_e : Number of neurons/bits (SDR dimensionality).

ω_ν : Number of active bits (e.g., $\|\nu\|_1$).

ς : Sparsity ratio.

c_ν : SDR capacity (combinatorial count).

List of Abbreviations

6-DoF : Six Degrees of Freedom.

ATE : Absolute Trajectory Error.

AUV : Autonomous Underwater Vehicle.

BA : Bundle Adjustment.

BoW : Bag-of-Words.

BRIEF : Binary Robust Independent Elementary Features.

CANN : Continuous Attractor Neural Network.

CNN : Convolutional Neural Network.

DBoW2 : Database of Bag-of-Words.

DR : Dead Reckoning.

DVL : Doppler Velocity Log.

EKF : Extended Kalman Filter.

EM : Experience Map.

FAB-MAP : Fast Appearance-Based Mapping.

FAST : Features from Accelerated Segment Test.

F1 : Harmonic mean of Precision and Recall (F_1 score).

FN : False Negative.

FP : False Positive.

GPS : Global Positioning System.

GRP : Gaussian Random Projection.

GT : Ground Truth.

HTM : Hierarchical Temporal Memory.

ID : Identifier.

IMU : Inertial Measurement Unit.

LBL : Long Baseline.

LC : Loop Closure.

LSBH : Locality Sensitive Binary Hashing.

LVC : Local View Cell.

LVC ID : Local View Cell identifier.

MAP : Maximum A Posteriori.

NED : North–East–Down frame convention.

NuPIC : Numenta Platform for Intelligent Computing.

ODOM : Odometry (used as reference such as MAP/ODOM proxies).

ORB : Oriented FAST and Rotated BRIEF.

ROS : Robot Operating System.

ROV : Remotely Operated Vehicle.

SAD : Sum of Absolute Differences.

SDR : Sparse Distributed Representation.

SE(3) : Special Euclidean group in 3D.

SIFT : Scale-Invariant Feature Transform.

SINS : Strapdown Inertial Navigation System.

SLAM : Simultaneous Localization and Mapping.

SO(3) : Special Orthogonal group in 3D.

SP : Spatial Pooler.

SV-CELLS : Spatial-View Cells

SURF : Speeded-Up Robust Features.

TM : Temporal Memory.

TP : True Positive.

VIA : Visual–Inertial–Acoustic.

VPR : Visual Place Recognition.

Chapter 1

Introduction

1.1 The Challenge of Long-Term Underwater Visual SLAM

The exploration and inspection of confined, GPS-denied underwater environments remain one of the most demanding frontiers in modern robotics [1]. Typical scenarios include submerged caves, shipwrecks, flooded mines, and complex industrial infrastructures, where operation is constrained by limited spatial range, overhead occlusions, obstacles, high turbidity, and poor ambient illumination [1]. In these environments, an Autonomous Underwater Vehicle (AUV) must maintain accurate navigation to ensure mission success and vehicle safety. This requires robust Simultaneous Localization and Mapping (SLAM), often under conditions where conventional positioning solutions are infeasible. For instance, acoustic infrastructure such as Long Baseline (LBL) arrays requires pre-deployment of beacons, while geophysical matching approaches depend on prior environmental maps; both are frequently impractical in exploratory or time-critical operations [1].

AUVs, therefore, rely on onboard navigation pipelines that combine self-motion estimation (e.g., Dead Reckoning—DR—and Strapdown Inertial Navigation Systems—SINS) with exteroceptive sensing. However, DR and inertial navigation inevitably suffer from unbounded drift, which can accumulate to mission-limiting errors in long traverses. This motivates the use of perception-driven correction mechanisms, notably loop closure and place recognition, as the core tools to constrain drift over time [2]. Visual SLAM has consequently become a research hotspot due to the low cost, low power consumption, and rich information content of cameras [2]. Nevertheless, the underwater domain imposes unique stressors that severely impair the visual sensing chain: "scattering and absorption" lead to strong attenuation and reduced contrast, frequently yielding blue-green shifted imagery [3]; furthermore, underwater scenes often exhibit repetitive structures or low texture, which

makes feature detection and matching unreliable [3]. While such degradation can be tolerable in open-water navigation with large safety margins, it is catastrophic in confined environments, where small localization failures can produce collisions or mission aborts [1].

The core long-term challenge, therefore, is to design a SLAM system that remains coherent in the presence of severe perceptual degradation and persistent perceptual aliasing, and that can maintain consistency across repeated traversals and extended missions.

1.2 Limitations of State-of-the-Art Approaches

Geometric Visual SLAM. The geometry-based paradigm is represented in this dissertation by ORB-SLAM3, a leading feature-based system that provides an open-source solution for visual, visual-inertial, and multi-map SLAM [4]. ORB-SLAM3 integrates short-, mid-, and long-term data association mechanisms and reuses past information to reduce drift in previously mapped areas. Its Atlas multi-map system improves robustness to tracking loss by spawning new maps and merging them when revisiting known places [4]. Despite this sophistication, its core reliance on robust matching of discrete ORB features becomes a critical point of failure in underwater environments characterized by low texture, repetitive patterns, or substantial appearance changes. This vulnerability is acknowledged by the authors: "The main failure case of ORB-SLAM3 is low-texture environments" [4]. In our experiments, under the repeated-traversal conditions simulated from the Underwater Cave Experiment [5, 6], ORB-SLAM3 collapses in its monocular-inertial configuration, failing to deliver a coherent long-term map. This result illustrates a fundamental limitation: sensor fusion can mitigate motion uncertainty, but it cannot compensate for a perceptual front-end that is brittle under severe aliasing.

Canonical bio-inspired SLAM. The bio-inspired paradigm offers an alternative route, exemplified by RatSLAM, a canonical framework modeled on rodent hippocampal navigation [7, 8]. RatSLAM maintains multiple pose hypotheses in a Continuous Attractor Network (CANN) of pose cells, enabling competition between hypotheses until sensory evidence resolves ambiguity. Its architecture combines (i) pose cells, (ii) local view cells for appearance recognition, and (iii) an experience map that builds a globally consistent topological representation [8]. While this design confers robustness to short-term ambiguity, RatSLAM's original visual front-end relies on low-resolution template matching (e.g., Sum of Absolute Differences—SAD), which is ill-suited for severe appearance changes and long-term underwater degradation [8]. Our experiments confirm this vulnerability: on the same challenging

dataset, RatSLAM fails to sustain reliable Visual Place Recognition , causing the experience map to collapse into a topologically incoherent graph.

Collectively, these results indicate that, under sustained perceptual aliasing and long-term appearance variability, the dominant failure mode shifts from back-end graph optimization to the reliability of the *perceptual data-association front-end*. In particular, both (i) discrete feature correspondences (ORB) and (ii) holistic template similarity (SAD) are insufficient to provide stable place evidence for long-term loop closure, which in turn degrades topological consistency. This dissertation addresses this limitation through the development of **PoseidonSLAM**, which casts perceptual disambiguation as **temporal sequence verification** using **sequence-aware neocortical representations**. Moreover, a **VPR Guard** mechanism regulates topological event emission based on temporal consistency, reducing spurious associations and improving map-graph stability within a unified Visual-Inertial-Acoustic Underwater SLAM framework.

1.3 NeoSLAM: A Neocortical-Inspired Paradigm for Perceptual Robustness

To overcome the dual failure identified above, this dissertation posits a shift away from static feature matching and toward sequence-grounded perception inspired by biological cognition. The motivation is consistent with the observation that "even animals with very small brains excel at combining local visual cues and self-motion cues for spatial navigation," suggesting that biologically inspired SLAM remains a promising path for robust navigation [2]. In the underwater domain, this direction has gained momentum as an alternative to classical filtering and optimization pipelines [3].

Within this context, this dissertation introduces and validates NeoSLAM, a novel neocortex-inspired Visual SLAM method designed for robust long-term performance by integrating computational models of cortical sequence processing [9, 10]. NeoSLAM represents a conceptual departure from hippocampus-centered approaches such as RatSLAM: rather than relying on instantaneous template similarity, NeoSLAM models place recognition as a *temporal inference* problem, where the identity of a location is supported by consistent transitions over time.

NeoSLAM is grounded in the Hierarchical Temporal Memory framework and employs Sparse Distributed Representations to encode visual information [9]. SDRs are high-dimensional binary vectors with high representational capacity and strong tolerance to noise, occlusion, and partial corruption. Instead of matching sparse keypoints (as in ORB-SLAM3) or raw templates (as in RatSLAM), NeoSLAM learns

and recognizes *sequences of SDRs*. This sequence-based memory provides contextual disambiguation, directly targeting the dominant failure mode of low-texture and perceptually ambiguous environments [4]. Moreover, the SDR/HTM mechanism enables an efficient, neuroscience-inspired loop-closure detector suitable for real-time execution on resource-constrained robotic platforms [9].

1.4 From NeoSLAM to PoseidonSLAM: Toward Visual-Inertial-Acoustic Underwater SLAM

While NeoSLAM establishes the core principle of neocortical sequence verification for robust place recognition, real underwater autonomy typically requires multimodal state estimation. In practice, inertial and acoustic streams (e.g., IMU, DVL, pressure/depth, and acoustic ranging when available) provide essential motion priors, scale observability, and navigation continuity when vision degrades. In such Visual–Inertial–Acoustic (VIA) pipelines, the role of the visual subsystem is not merely to estimate motion, but to deliver *high-integrity revisitation constraints* (loop closures) that remain reliable under aliasing.

To bridge the gap between NeoSLAM as a robustness mechanism and VIA underwater navigation as a system requirement, this dissertation introduces **PoseidonSLAM** as a system-level realization that operationalizes robust visual place recognition and its controlled injection into a topological mapping process. PoseidonSLAM incorporates: (i) a NeoSLAM-aligned VPR pathway, (ii) an auditable event–evidence association mechanism (ensuring traceability between "experience events" and their perceptual evidence), and (iii) a **VPR Guard** gating strategy that regulates experience creation under ambiguous perceptual conditions. This design is motivated by a key observation: in multimodal fusion, false visual constraints can be more damaging than missing ones, because incorrect loop closures can deform the global estimate and corrupt map consistency. Therefore, the system must not only detect revisitation, but also *control when* perceptual constraints are allowed to influence the global representation.

The PoseidonSLAM results reported in this dissertation provide solid evidence aligned with VIA Underwater SLAM requirements: the system supports controlled long-term mapping through (a) regulated event density, (b) stable graph growth behavior, and (c) explicit auditing of event-to-evidence synchronization. In addition, PoseidonSLAM is evaluated through stability-oriented metrics and visualizations that characterize map deformation proxies (e.g., span and volume ratios) and graph structure indicators (nodes, edges, mean degree), complementing classical localization metrics when appropriate. Collectively, these artifacts establish PoseidonSLAM

as a robust stepping stone from NeoSLAM's neocortical principle toward reliable Visual–Inertial–Acoustic underwater SLAM for autonomous-operation pipelines.

1.5 The POSEIDON NAV Project

The objective of this dissertation is threefold:

1. **Validate NeoSLAM for underwater visual robustness:** empirically demonstrate that a neocortical-inspired, sequence-based SLAM framework can sustain coherent mapping in a regime where a state-of-the-art geometric baseline (ORB-SLAM3) and a canonical bio-inspired baseline (RatSLAM) fail.

2. **Identify an efficient and effective visual front-end for robust VPR:** perform a controlled comparison between two pre-trained CNN architectures—AlexNet-conv3 and ShuffleNetV2 x1.0—to determine which offers the best robustness-to-cost trade-off under underwater perceptual degradation.

3. **Introduce PoseidonSLAM and define POSEIDON NAV as the ROS-based system architecture and experimentation stack for 3D multimodal VIA-SLAM:** operationalize robust VPR within a system-level pipeline and present POSEIDON NAV as the ROS-based architectural blueprint and experimental framework within which the PoseidonSLAM method is instantiated, with explicit alignment to Visual-Inertial-Acoustic underwater operation.

To achieve these goals, we designed an *instrumented and reproducible* evaluation workflow within the Robot Operating System (ROS) framework [11], comprising standardized data ingestion, controlled ablation runs, and post-run diagnostic audits that *verify* event–evidence traceability and quantify map-graph behavior through topological stability *proxies*. The evaluation uses an *Underwater Cave Transformed Database* derived from the public Underwater Cave Experiment Dataset [5, 6] to isolate and stress-test VPR robustness under controlled appearance and viewpoint perturbations, while PoseidonSLAM is additionally assessed on the original benchmark to characterize the system-level Visual-Inertial-Acoustic Underwater SLAM framework. This setting was used to: (i) characterize baseline failures, (ii) validate NeoSLAM robustness, and (iii) quantify PoseidonSLAM behavior under controlled ablations (notably VPR Guard ON vs. OFF).

The main contributions of this dissertation are:

- **A failure-grounded benchmark protocol for long-term visual underwater SLAM:** a repeated-traversal evaluation protocol derived from a real

underwater cave experiment [5, 6], designed to *systematically elicit* perceptual-aliasing failure modes and to support controlled robustness analysis.

- **Underwater validation of NeoSLAM as a neocortical-inspired robustness mechanism:** empirical evidence that HTM-based *sequence processing* sustains coherent topological mapping under severe aliasing regimes [9], where representative geometric [4] and canonical bio-inspired [8] baselines lose coherence.

- **A controlled CNN front-end study for robust VPR under resource constraints:** a comparative assessment showing that ShuffleNetV2 x1.0 yields a superior robustness–efficiency trade-off relative to a legacy baseline (AlexNet-conv3), supporting real-time deployment considerations for underwater platforms [12–14].

- **PoseidonSLAM as unified Visual-Inertial-Acoustic Underwater SLAM framework with topological graph stabilization:** the development of PoseidonSLAM as a system-level integration that couples neocortical-inspired, sequence-aware VPR with 3D topological mapping principles, targeting robust VIA navigation in GPS-denied underwater scenarios. The framework further introduces (i) a *VPR Guard* mechanism to regulate topological event emission based on temporal consistency, and (ii) an event–evidence audit procedure that enforces traceability between topological events and sensory evidence, enabling reproducible ablations and quantitative reporting of long-term mapping stability in an underwater trajectory scenario.

1.6 Structure of This Dissertation

This dissertation is organized into six chapters:

Chapter 1 — Introduction: Presents the challenges of underwater localization, establishes the scientific gap by demonstrating the failure of both geometric and canonical bio-inspired SLAM under perceptual aliasing, and introduces NeoSLAM and PoseidonSLAM as steps toward multimodal robustness.

Chapter 2 — Review of Underwater Visual SLAM: Surveys the state-of-the-art, categorizing existing techniques and identifying the limitations that motivate neocortical-inspired sequence verification and controlled perceptual constraint injection.

Chapter 3 — Theoretical Foundations: Details the theoretical principles underpinning ORB-SLAM3, RatSLAM, and related bio-inspired systems, and formalizes the HTM/SDR foundations that motivate NeoSLAM.

Chapter 4 — The POSEIDON NAV Project: Introduces the POSEIDON NAV architecture and situates PoseidonSLAM within the broader goal of Visual-Inertial-Acoustic Underwater SLAM in 3D, emphasizing modularity and robustness.

Chapter 5 — Experimental Validation and Results: Describes the experimental setup and transformed dataset, reports comparative failures of ORB-SLAM3 and RatSLAM, validates NeoSLAM performance and CNN front-end selection, and presents PoseidonSLAM results and ablations through stability-oriented metrics and visual evidence.

Chapter 6 — Conclusion: Synthesizes the findings, discusses limitations, and outlines a research agenda toward field-ready, lifelong, Visual–Inertial–Acoustic underwater autonomy.

Appendix A — MATLAB Script for Experimental Processing: Documents the analysis scripts used for post-processing, plotting, and metric extraction.

Chapter 2

Review of Underwater Visual SLAM

To justify the methodological approach of this dissertation, it is essential to first conduct a critical analysis of the state-of-the-art in Underwater Visual SLAM. This chapter moves beyond a simple survey to build a clear argument: while significant progress has been made, conventional SLAM paradigms exhibit fundamental limitations in long-term, dynamic underwater operations. This chapter argues, based on the literature and on the operational requirements of long-term underwater navigation, that the dominant SLAM pipelines remain limited primarily by perceptual robustness and data-association reliability under turbidity, low texture, and structural repetition. These constraints motivate the investigation of neuro-inspired, sequence-aware place-recognition mechanisms and their integration into a multimodal Visual-Inertial-Acoustic Underwater SLAM framework, as later instantiated by PoseidonSLAM. The analysis will follow a logical progression, starting with the vulnerabilities of the classic pipeline and culminating in the identification of the research frontier where this work is positioned.

2.1 Related Works

The conventional Underwater Visual SLAM framework is a structured pipeline, typically comprising a **Front-end** for state estimation, a **Back-end** for optimization, and a **Loop Closure** module for map correction [3]. While effective in controlled settings, each component becomes a point of failure when subjected to the unstructured and noisy conditions of the underwater domain.

The **Visual Front-end**, responsible for odometry through feature tracking (using algorithms like SIFT, SURF, or ORB [15–17]), is the first and most frequent point of failure. Light attenuation, backscatter from suspended particles, and low-texture seabeds severely degrade image quality, leading to unreliable feature detection and the intractable problem of data association [3, 18]. Although image enhancement and restoration techniques exist [19, 20], they are often computationally

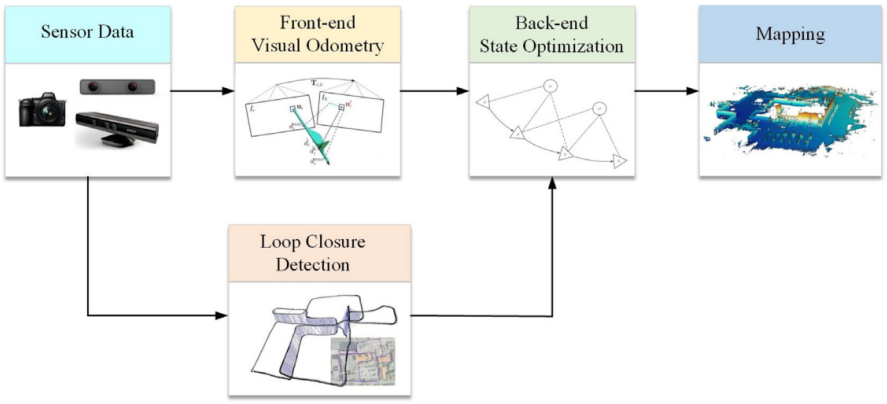


Figure 2.1: The conventional Underwater Visual SLAM framework, illustrating the key modules from sensor data acquisition to final mapping. While logically structured, each component faces unique robustness challenges in aquatic environments. Source: Reproduced from [3].

expensive and insufficient for the monotonic and feature-scarce scenes common in underwater exploration. Complementary robustness measures have therefore emerged that explicitly modulate the influence of visual information under degradation. VIA-SLAM, for example, introduces an adaptive dynamic weighting factor driven by the trend of RMS reprojection residuals and the number of successfully tracked feature points, reducing the impact of poor image quality by shifting reliance toward IMU and DVL constraints when vision becomes unreliable [21].

The **Back-end** is tasked with optimizing the vehicle's trajectory and map by minimizing accumulated errors. Early approaches relied heavily on **probabilistic filtering** (e.g., EKF, UKF, PF) [22–24], but these methods are computationally demanding and prone to catastrophic divergence on long-term missions, as linearization errors accumulate irrevocably [25]. This led to the dominance of **graph-based optimization**, which uses historical data to refine the entire trajectory [26]. Frameworks like ORB-SLAM [27] and its variants represent the state-of-the-art in pose-graph optimization [28, 29]. However, their performance is critically dependent on the third component: loop closure.

Loop Closure detection, the ability to recognize a previously visited place to reset accumulated drift, is the Achilles' heel of traditional SLAM in dynamic environments. Methods based on appearance, such as Bag-of-Words (BoW) [30] or hashing [31], struggle immensely with the perceptual aliasing typical of underwater scenes (e.g., similar-looking coral reefs or cave walls). A single false positive loop closure can corrupt the entire map, while frequent false negatives prevent the system from correcting drift at all [3]. While deep learning approaches have shown promise in generating more robust image descriptors [32, 33], they do not fundamentally solve the problem of reasoning under severe and long-term appearance variation.

The inherent fragility of this classic pipeline in underwater environments is not merely theoretical. In a comparative study conducted by QIN *et al.* [2], state-of-the-art geometric methods like ORB-SLAM2, despite exhibiting the highest metric accuracy on favorable sequences, showed tracking failures in segments with abrupt movements or overexposure. This empirically validates our assertion that optimizing for precision in classic methods often comes at the expense of **robustness**—the most desired characteristic for autonomous navigation in unpredictable scenarios.

When loop closure is unlikely or impossible (e.g., long transects without revisiting), an alternative is to inject absolute acoustic constraints that curb global drift without appearance-based relocalization. Acoustic-VINS illustrates this direction by tightly coupling LBL slant-range measurements with visual and inertial factors and refining the solution via a coarse-to-fine strategy that includes global pose graph optimization with LBL measurements [34].

2.2 Evolution of Methodologies and Persistent Gaps

In response to the pipeline's vulnerabilities, research has evolved in several directions. However, as we will argue, these advances, while valuable, constitute incremental improvements rather than a fundamental solution to the long-term autonomy problem.

2.2.1 Multi-Sensor Fusion

A primary strategy to mitigate visual data unreliability is **multi-sensor fusion** [35, 36]. By integrating data from proprioceptive sensors like IMUs and DVLs, and exteroceptive sensors like sonar, systems can maintain localization during periods of visual failure [37]. The SVIn2 framework, for example, demonstrated state-of-the-art accuracy by tightly coupling sonar, visual, inertial, and depth data [38]. Similarly, magnetometer data has been used to resolve orientation drift in underwater caves [39].

Recent DVL-integrated pipelines further show that proprioceptive acoustic cues can directly address monocular scale ambiguity and fragile initialization: VIA-SLAM proposes a DVL-assisted coarse-to-fine initialization to estimate scale and gravity-related parameters, and couples DVL velocity residuals with inertial preintegration and visual reprojection residuals in a factor-graph backend, while dynamically adjusting the weight of visual information [21]. In contrast, LBL-based systems provide global anchoring: Acoustic-VINS incorporates LBL slant-range factors and

performs global pose-graph refinement with LBL measurements to obtain globally consistent trajectories in large-scale scenarios without loop closure [34].

Critical Gap: While essential, sensor fusion is a **compensatory mechanism**, not a solution to the core problem of **Visual Place Recognition**. It provides better dead-reckoning between visual anchor points but does not inherently improve the robot's ability to recognize a place when its appearance has changed. Without robust VPR, even a multi-sensor system will eventually fail on long-term or multi-session missions [40, 41].

2.2.2 Deep Learning and Feature-based Descriptors

Recognizing the limitations of handcrafted features, the community has turned to deep learning for more robust VPR. This includes end-to-end visual odometry networks [42] and advanced descriptors for loop closure [43]. Recent work has focused on creating "universal" place descriptors that are invariant to viewpoint and appearance changes. The tutorial by SCHUBERT *et al.* [44] provides a comprehensive overview of VPR challenges, while frameworks like AnyLoc have shown impressive gains by using foundation models like DINOv2 to generate robust, general-purpose features [45].

Critical Gap: Despite their power, these methods often require vast amounts of training data, and their "universality" is not guaranteed in the highly specific and challenging underwater domain. More importantly, they treat place recognition as a one-shot image retrieval problem, largely ignoring the **temporal context** of navigation—a key element that biological systems use to disambiguate perceptually similar locations.

2.3 The Bio-Inspired Approach

The persistent gaps in long-term robustness have motivated a third, more radical approach: modeling the neural mechanisms of animal navigation. This paradigm can be technically framed within the domain of **Spiking Neural Networks (SNNs)**, the class of models that most closely approximates the function of biological neurons. As analyzed by QIN *et al.* [2], SNNs represent a promising frontier for SLAM as they focus on event-driven and asynchronous information processing, ideal characteristics for handling noisy and sporadic sensory data.

Pioneering systems like **RatSLAM** [46–48], were the first to demonstrate the viability of this approach, shifting the focus from geometric precision to topological consistency and resilience over time. Adaptations for the underwater domain, such as **DolphinSLAM** [49] and **Hippo 3D** [50], extended these concepts to 3D

environments, showing potential in visually feature-scarce scenarios. Notably, in the comparative study by QIN *et al.* [2], RatSLAM, while not the most metrically accurate, demonstrated stable and robust navigation, reinforcing the idea that the bio-inspired/SNN paradigm prioritizes **resilience** over optimized precision.

Unlike traditional methods, these systems do not rely on matching discrete features. Instead, they use continuous attractor networks and sequence-based memory to represent the robot's pose and experience. This approach is inherently more robust to the kind of gradual and significant appearance changes seen underwater. More recent frameworks like BioSLAM explicitly model lifelong memory systems to balance learning and retention [51].

This paradigm directly addresses the limitations we identified:

- It handles **perceptual degradation** not as a failure case, but as a normal operating condition.
- It leverages **temporal context** to disambiguate locations, a feature absent in most deep learning-based VPR systems.
- It prioritizes **long-term map coherence** over short-term metric accuracy, which is crucial for sustained autonomy.

This body of work [2, 48, 52] strongly suggests that the next leap in underwater navigation will come from more sophisticated brain-based models. It is precisely at this frontier that our research is situated. We hypothesize that by leveraging a more advanced computational model of the neocortex, as embodied by NeoSLAM, we can achieve a new level of performance in long-term Underwater Visual SLAM.

Chapter 3

Theoretical Foundations

This chapter presents the theoretical foundations underlying the localization and mapping approaches investigated in this dissertation. The discussion is structured around biologically inspired SLAM systems, which draw motivation from neural mechanisms observed in mammals, and a state-of-the-art geometric SLAM system employed as a comparative benchmark.

The chapter begins with RatSLAM, a seminal hippocampus-inspired framework that models spatial cognition through continuous attractor dynamics and topological mapping. RatSLAM establishes the biological basis for representing space using neural populations rather than explicit geometric reconstruction, emphasizing robustness to perceptual ambiguity and odometric drift.

Subsequently, DolphinSLAM is introduced as an extension of the RatSLAM paradigm to three-dimensional underwater environments. DolphinSLAM incorporates multimodal sensing and probabilistic place recognition, demonstrating how hippocampal-inspired architectures can be adapted to volumetric navigation under severe perceptual degradation.

Building upon these foundations, NeoSLAM is presented as a neocortex-inspired evolution of the RatSLAM architecture. NeoSLAM integrates deep visual representations and Hierarchical Temporal Memory (HTM) models to enhance long-term place recognition under drastic appearance changes, while preserving the robust topological back-end inherited from hippocampal models.

Finally, ORB-SLAM3 is introduced as a geometric, optimization-based SLAM system serving as a benchmark for comparison. Unlike the biologically inspired approaches discussed earlier, ORB-SLAM3 relies on explicit feature extraction, probabilistic state estimation, and bundle adjustment to construct metrically accurate maps. Its inclusion enables a rigorous quantitative and qualitative comparison between bio-inspired SLAM systems and a state-of-the-art geometric baseline, highlighting the respective strengths, limitations, and failure modes of each paradigm.

3.1 RatSLAM

Within the context of this dissertation, RatSLAM constitutes the foundational biologically inspired SLAM framework upon which subsequent models are analyzed and extended.

RatSLAM is a landmark bio-inspired SLAM system that translates hippocampal navigation principles into a functional robotic SLAM architecture [7]. Conceptually, it decomposes long-term navigation into three tightly coupled subsystems: (i) a continuous pose-belief representation implemented as a competitive continuous attractor network (Pose Cells), (ii) a perceptual recognition layer that detects revisited scenes (Local View Cells), and (iii) a graph-based topological memory that consolidates loop closures into a globally interpretable map (Experience Map) [8].

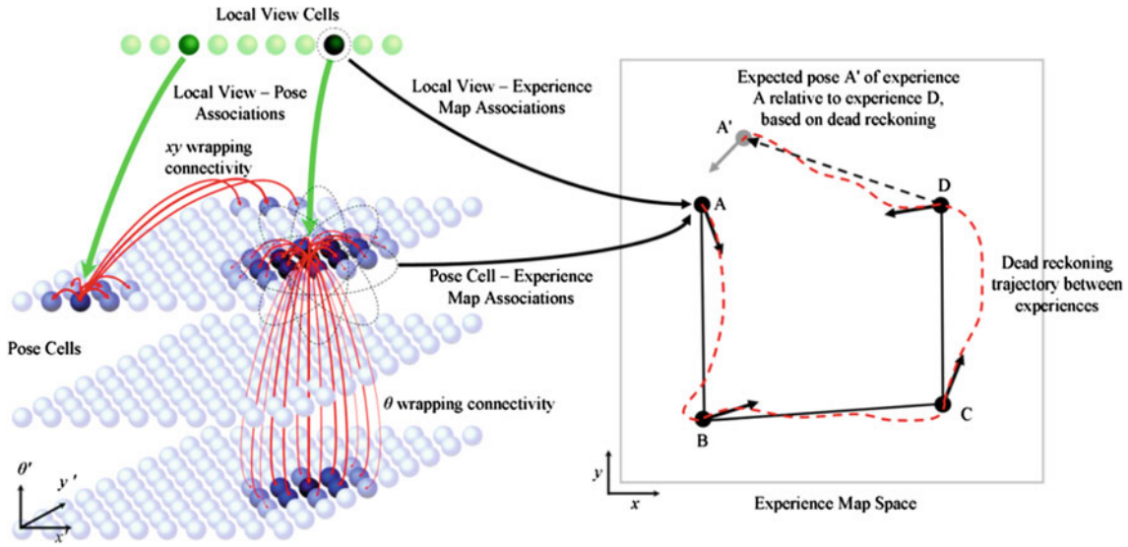


Figure 3.1: The architecture of the RatSLAM system, illustrating the interaction between Local View Cells (visual recognition), Pose Cells (pose estimation), and the Experience Map (topological memory). Source: Reproduced from [8].

3.1.1 Pose Cells

The Pose Cell module implements a three-dimensional Continuous Attractor Network (CAN) over (x, y, θ), maintaining the robot's pose belief as an *activity packet* whose dominant mode encodes the current pose estimate [8]. In the OpenRatSLAM formulation, the network is arranged as a prism with a square (x, y) plane of side length S_{xy} and a discretized heading axis with S_θ bins. Critically, recurrent connections *wrap across all six faces of the pose-cell network*, preserving uniform neighborhood structure at the boundaries and allowing the activity packet to translate/rotate without edge artifacts [8].

The attractor behavior is achieved via locally excitatory and globally inhibitory connectivity defined by the Mexican-hat (Difference-of-Gaussians) footprint $\varepsilon_{a,b,c}$:

$$\varepsilon_{a,b,c} = e^{-\frac{a^2+b^2}{k_p^{\text{exc}}}} e^{-\frac{c^2}{k_d^{\text{exc}}}} - e^{-\frac{a^2+b^2}{k_p^{\text{inh}}}} e^{-\frac{c^2}{k_d^{\text{inh}}}}, \tag{3.1}$$

where (a, b, c) denote inter-unit offsets along (x, y, θ). The constants k_p and k_d control the spatial spread of coupling in *place* and *direction*, respectively, and are typically treated as fixed (the reference implementation reports them as the result of extensive tuning and not requiring further adjustment) [8].

At each update, the internal recurrent input at unit (x', y', θ') is computed by summing the kernel-weighted activity of all units and subtracting a global inhibition term ϕ:

$$\Delta P_{x',y',\theta'} = \sum_{i=0}^{S_{xy}-1} \sum_{j=0}^{S_{xy}-1} \sum_{k=0}^{S_\theta-1} P_{i,j,k}\, \varepsilon_{a,b,c} \; - \; \phi, \tag{3.2}$$

where $P_{i,j,k}$ denotes the current activity of the Pose Cell located at lattice coordinates (i, j, k); (x', y', θ') is the target cell for which the recurrent update is being computed; S_{xy} is the side length of the square (x, y) lattice, and S_θ is the number of discrete bins along the heading axis (in OpenRatSLAM, $S_\theta = 36$, i.e., 10° per bin) [8]. The indices satisfy $i, j \in \{0, \ldots, S_{xy} - 1\}$ and $k \in \{0, \ldots, S_\theta - 1\}$.

The term $\varepsilon_{a,b,c}$ is the recurrent coupling (synaptic footprint) given by the Difference-of-Gaussians kernel in Eq. (3.1), evaluated at the wrapped lattice offsets (a, b, c) between the source cell (i, j, k) and the target cell (x', y', θ'). Concretely, a and b are the wrapped offsets along the two spatial axes and c is the wrapped offset along the heading axis; under wrap-around connectivity, these offsets are computed as minimum distances on a periodic lattice, so that interaction neighborhoods remain translation- and rotation-consistent when the activity packet crosses network boundaries.

Finally, ϕ is a global inhibition term applied uniformly to all cells, preventing unbounded growth of activation and enforcing competitive dynamics by suppressing weak or spurious activity relative to the dominant packet.

Operationally, Eq. (3.2) implements a wrapped 3D convolution of the current activity state with the kernel ε, followed by global inhibition; this mechanism preserves a single coherent activity packet under typical noise conditions.

Self-motion (odometry) shifts Pose Cell activity to reflect incremental motion, implementing *path integration* [8]. Visual recognition events, in turn, inject activity into the network, biasing the pose belief toward previously learned pose hypotheses and enabling loop closure. While typical operation yields a *dominant* packet (used for centroid extraction), RatSLAM's competitive attractor dynamics can transiently sustain competing hypotheses when perceptual evidence is ambiguous, until subse-

quent sensory input resolves the competition [7].

3.1.2 Local View Cells

This module serves as the system's perceptual front-end, learning and identifying unique visual scenes. It achieves this by converting camera images into low-resolution *visual templates* and comparing them against a database of previously seen templates using a Sum of Absolute Differences (SAD) metric. When a familiar scene is recognized, the corresponding Local View Cell (LVC) injects activity into the Pose Cell network, providing the mechanism for correcting odometric drift and closing loops.

Each LVC encodes a *visual template* formed by cropping, down-sampling and optionally normalizing the camera frame; recognition uses SAD match with horizontal shift compensation. Horizontal shift compensation accounts for rotational invariance in panoramic or forward-facing camera setups.

When a novel scene is encountered, a new Local View Cell is created and its template ID is associated (one-shot) with the centroid of the currently dominant pose-cell activity packet. Upon revisiting that scene, the corresponding Local View Cell reactivates and injects activity back into the pose-cell lattice at the previously learned location, providing the loop-closure mechanism that corrects accumulated path-integration drift.

$$\Delta P_{x',y',\theta'} = \delta \sum_i \beta_{i,x',y',\theta'} V_i, \tag{3.3}$$

In Eq. (3.3), δ controls the influence of visual evidence relative to odometry-driven path integration; $V_i \in \{0, 1\}$ indicates whether template i is currently active; and $\beta_{i,x',y',\theta'}$ denotes the learned excitatory link from Local View Cell i to the Pose Cell units near the pose centroid observed when that template was first created. In practice, repeated matches of the same template are subject to a rapid saturation/decay mechanism, preventing spurious re-localizations when the platform is stationary for extended periods.

3.1.3 Experience Map

Since the Pose Cell network is finite and its dimensions wrap around, a single Pose Cell can represent multiple physical locations. The Experience Map resolves this ambiguity by creating a *topological graph* where each node, or experience, links a Local View Cell activation to its corresponding Pose Cell state. A graph relaxation algorithm then distributes error throughout the map after a loop closure, ensuring the creation of a globally consistent spatial memory.

To disambiguate the finite Pose Cell lattice and distribute drift, the Experience

Map stores nodes e_i that bind concurrent Pose Cell and Local View Cell codes to a 2D map coordinate:

$$e_i = \{\, P^i,\ V^i,\ \mathbf{p}^i \,\}. \tag{3.4}$$

When deciding whether to create a new node, the current states (P, V) are compared to each experience e_i via the score metric

$$S_i = \mu_p \,|P_i - P| + \ \mu_v \,|V_i - V| \tag{3.5}$$

where μ_p and μ_v weight the respective contributions of Pose Cell and Local View states to the matching score. If $\min(S_i) \geq S_{\max}$, a new experience is created. As the agent moves, directed links are added

$$\ell_{ij} = \left\{\Delta\mathbf{p}^{ij},\ \Delta t^{ij}\right\}, \tag{3.6}$$

storing relative odometry and traversal time (used for quickest-path planning with Dijkstra). After loop closure, the odometric error is spread by graph relaxation:

$$\Delta\mathbf{p}^{i} = \alpha \left[\sum_{j=1}^{N_f} \left(\mathbf{p}^{j} - \mathbf{p}^{i} - \Delta\mathbf{p}^{ij}\right) + \sum_{k=1}^{N_t} \left(\mathbf{p}^{k} - \mathbf{p}^{i} - \Delta\mathbf{p}^{ki}\right) \right], \tag{3.7}$$

with rate $\alpha \in (0, 1)$, N_f links leaving e_i, and N_t links entering e_i.

The primary strength of the RatSLAM architecture is its inherent robustness to perceptual ambiguity and odometric error. Through a competitive continuous attractor network, RatSLAM maintains a distributed pose belief in which alternative hypotheses can coexist and compete until sensory evidence resolves the ambiguity [7]. These hypotheses compete until subsequent sensory evidence reinforces the correct one, allowing the system to recover from both minor and major path integration errors. This theoretical robustness provides the foundation upon which practical systems can be built.

Understanding these core modules—particularly the Pose Cells and the Experience Map-is essential as they form the architectural backbone that NeoSLAM adopts and enhances with a neocortical front-end.

Didactic walk-through: a single loop-closure over four experiences

To make RatSLAM's operational logic explicit, consider Fig. 3.1 and a traversal $A \to B \to C \to D$ followed by a revisit of A. Assume odometry accumulates drift, so dead reckoning after the circuit does not coincide exactly with the initial pose at A.

1. **First visit to A: template creation and pose encoding.** A novel scene at A creates a new Local View Cell (e.g., $V_1 = 1$) and stores its visual template. Concurrently, the Pose Cells encode the current belief as an activity packet with centroid (x_A, y_A, θ_A).

2. **Why a packet persists (recurrent competition).** The Mexican-hat kernel $\varepsilon_{a,b,c}$ (Eq. (3.1)) implements local excitation and distal inhibition. Applied through the recurrent update (Eq. (3.2)) with global inhibition ϕ, these dynamics stabilize coherent belief modes and suppress spurious competitors under typical noise levels.

3. **From A to B: path integration.** As the robot moves, odometry shifts pose-cell activity to represent incremental motion, moving the packet centroid toward (x_B, y_B, θ_B) while accumulating drift over time.

4. **Experience creation and linking (persistent topology).** The Experience Map stores experiences $e_i = \{P_i, V_i, \mathbf{p}_i\}$ (Eq. (3.4)). As the traversal proceeds, the system either reuses an experience or creates a new one using the score in Eq. (3.5). When a new node is created, a directed link $\ell_{ij} = \{\Delta\mathbf{p}^{ij}, \Delta t^{ij}\}$ (Eq. (3.6)) is added to store relative motion and traversal time.

5. **Revisiting A: perception initiates loop closure (local cue).** Near A, the current view matches the stored template, reactivating V_1. Activity injection (Eq. (3.3)) adds excitation at the pose-cell region associated with V_1 when it was first learned. In the reference system, each template injects only briefly (saturation), so reliable re-localization is reinforced when a sufficiently long sequence of familiar scenes is observed in the correct order, producing sustained injection that draws the dominant packet back toward the original hypothesis for A.

6. **Global coherence: relaxation distributes loop-closure residual.** After revisiting A, the EM contains a cycle whose accumulated odometry does not perfectly close. Graph relaxation (Eq. (3.7)) distributes this residual over linked experiences (e.g., along A–B–C–D), enforcing *global* topological coherence. In contrast, Pose Cells preserve *local* coherence by stabilizing the belief packet and integrating corrections from odometry (path integration) and perception (activity injection).

This walk-through makes the mechanism transparent: Local View recognition triggers loop closure; Pose Cells implement local competitive belief dynamics; and the Experience Map consolidates revisitation evidence into a globally coherent topological graph via relaxation.

3.1.4 OpenRatSLAM

This subsection focuses on implementation aspects relevant for reproducibility and experimental alignment with this dissertation.

OpenRatSLAM operationalizes RatSLAM as a modular, reproducible, and dataset-agnostic ROS system while preserving the canonical triad of Local View, Pose Cells, and Experience Map [8]. The following subsections summarize the implementation as used in this dissertation.

Visual Odometry

The optional *Visual Odometry* node estimates translational and rotational velocities from camera imagery and supplies odometry to the Pose Cell network. Its configuration exposes cropping windows for translation and rotation estimation, the horizontal camera field of view and frame rate, and scaling/limits on the reported translational speed. These parameters allow platform-specific calibration while keeping the downstream nodes agnostic to the odometry source (e.g., wheel or vision) [8].

Local View Match

The *Local View* node converts each frame into a low-resolution *visual template* and performs matching via sum-of-absolute-differences with optional horizontal shift compensation (disabled in panoramic mode). Images can be cropped to suppress non-salient regions (e.g., floor/road/water column), and template sizes are set explicitly. A sensitivity threshold governs whether a frame is considered novel or matched to an existing template; global and patch normalization options mitigate illumination variation and emphasize informative details. A recognized template injects activity into the Pose Cells at the previously learned association, enabling relocalization and loop closure [8].

Pose Cell Network

The *Pose Cells* node receives asynchronous inputs from odometry and Local View. For a new template, its ID is bound to the centroid of the current activity packet; for a recognized template, activity is injected at the stored location with rapid saturation and slow restoration to avoid spurious relocalizations while stationary. Each update cycle applies local excitation and inhibition, global inhibition, energy normalization, path integration (lattice shift using odometry, and centroid extraction, after which a topological action is decided for the Experience Map [8].

Experience Map

The *Experience Map* creates/updates a topological graph whose nodes store a pose estimate and orientation, and whose edges store the relative transform and traversal time derived from odometry. On every action (create node, create edge, set node), the map performs graph relaxation (Eq. 3.7) to distribute drift and maintain global consistency. The node publishes (i) the full topological map, (ii) the robot's current pose in map coordinates, and (iii) rviz markers; it also accepts a 2D goal and returns a quickest path computed over edge times [8].

Visualization

OpenRatSLAM provides live views of Local View matching, the Experience Map, and the Pose Cell activity packet (Fig. 4), and integrates with rviz to render pose and the topological map. For introspection, rxplot can monitor template/experience growth to diagnose false loop closures. Offline analysis scripts export topics from bag files and generate MATLAB plots and frame-match visualizations for quantitative assessment [8].

Software Architecture

OpenRatSLAM ROS Architecture, shown in Figures 3.2 and 3.3, is decomposed into four ROS nodes that execute as a continuous pipeline and communicate asynchronously via messages: *Local View Cells*, *Pose Cell Network*, *Experience Map*, and an optional *Visual Odometry* node used for image-only datasets. This split improves modularity and allows each node to run in a separate process and core; importantly, in this implementation the Pose Cell node decides *when* to create new Experience Map nodes/links because that decision depends on internal pose–packet dynamics that are not visible outside the class once modules are separated [8].

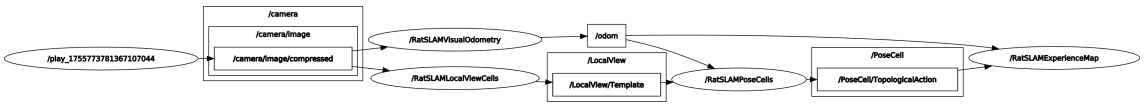


Figure 3.2: Visual Odometry OpenRatSLAM ROS Graph to *St Lucia 2007* Dataset. Source: Author's own creation based on [8].

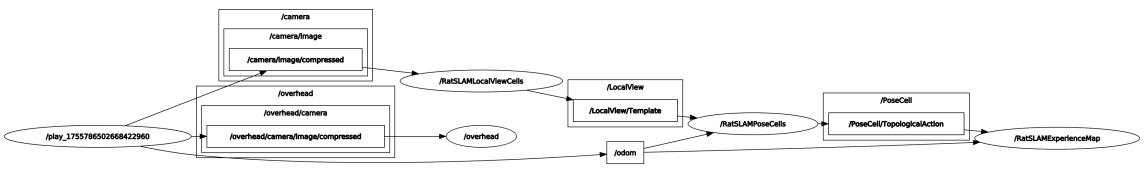


Figure 3.3: Visual-Inertial OpenRatSLAM ROS Graph to *iRat 2012* Dataset. Source: Author's own creation based on [8].

Each algorithmic class (`LocalViewMatch`, `PosecellNetwork`, `ExperienceMap`) is *independent of ROS* and compiled standalone; thin ROS callback wrappers provide I/O while keeping classes free of ROS dependencies. This design eases reuse with other packages and supports both online and offline operation [8].

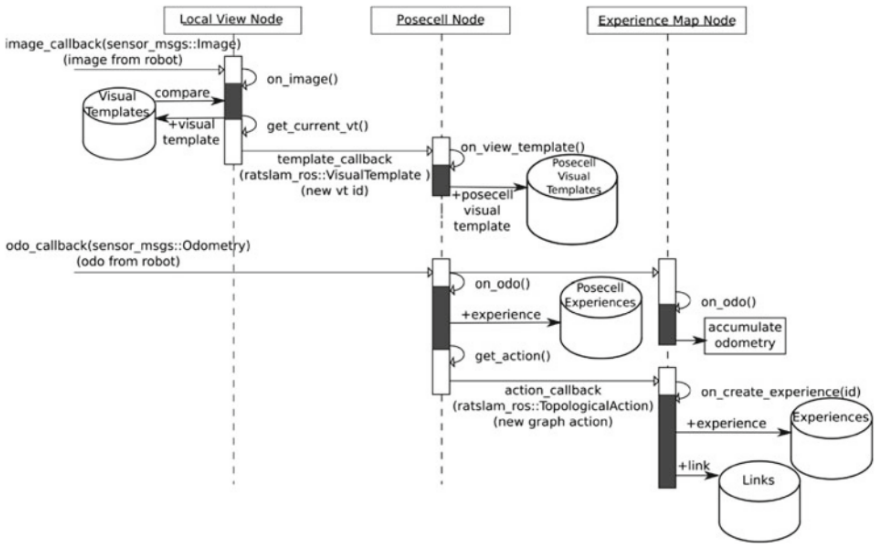


Figure 3.4: Sequence diagram for *new experience* creation: LV generates a template event; PC integrates odometry, injects visual evidence, and issues a `TopologicalAction`; EM creates node+link and relaxes the graph. Source: Reproduced from [8].

The sequence begins with incoming `sensor_msgs/Image` and `sensor_msgs/Odometry`:

1. **Local View node** (`image_callback`) forms a visual template and either *matches* or *creates* a new template; the event is sent as `ratslam_ros/VisualTemplate` (template ID). The node maintains a template store used for future recall.

2. **Pose Cell node** (`on_view_template`) recalls the LV–PC association and injects energy at the stored pose; (`on_odo`) shifts the packet by path integration; the centroid is extracted and the node selects a topological action: `CREATE_NODE`, `CREATE_EDGE`, or `SET_NODE`. The action is published as `ratslam_ros::TopologicalAction` with fields `src_id`, `dest_id`, and `relative_rad` [8].

3. **Experience Map node** consumes the action, accumulates odometry, creates the new *Experience* (and implicit link from the previously active node), and relaxes the graph; nodes and links are stored in efficient separate vectors [8].

When the current view *matches* a previous template, the PC recalls the associated experience; depending on the motion history and thresholds, the PC node

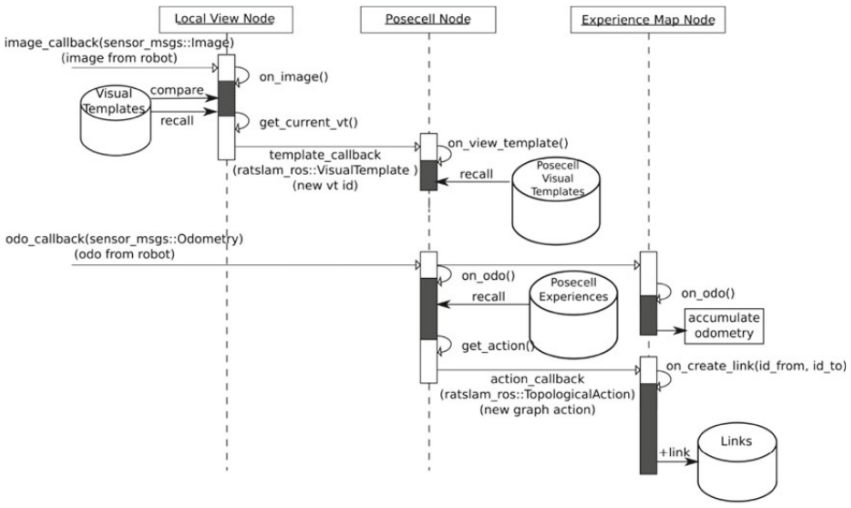


Figure 3.5: Sequence diagram for *new link* between existing experiences following recognition of a learnt template; note that recognition does not necessarily imply a link. Source: Reproduced from [8].

may issue a `CREATE_EDGE` action to connect the currently active experience to the recalled one. Note that matching a learnt template *does not guarantee* a new link; this prevents spurious connectivity during brief re-entries [8].

The EM node publishes: (i) a `TopologicalMap` containing arrays of `TopologicalNode` and `TopologicalEdge`, (ii) the robot pose in map coordinates, and (iii) `Marker` messages for `rviz`. It also computes a *quickest path* to a user goal (rviz 2D nav goal) and returns a `Path` message. Offline scripts export bag data and provide MATLAB tools (`show_em.m`, `plot_matches.m`, etc.) for auditing false matches and map quality [8].

For image-only datasets, a simple scanline-based VO estimates rotational velocity via horizontal offset minimizing mean absolute difference between consecutive profiles; translational speed scales with the minimum difference and is clamped to reject illumination shocks. Other nodes are agnostic to odometry source [8].

OpenRatSLAM: Code Specifications

OpenRatSLAM defines explicit messages to expose internal state and actions. The Pose Cell node emits a `ratslam_ros::TopologicalAction` with actions `CREATE_NODE`, `CREATE_EDGE`, or `SET_NODE`, along with source/destination IDs and relative rotation. The Experience Map node publishes a `TopologicalMap` (nodes and edges), the robot's pose in map space, and `Marker` messages for rviz rendering; edges carry transform and duration fields to support quickest-path queries. Wrappers and message designs enable both online operation and offline reproducibility via bag files [8].

OpenRatSLAM exposes a concise parameter set. Years of tuning yielded stable Pose Cell dynamics; in typical use only the Local View match threshold and, in some cases, the LV→PC injection gain need adjustment, while the remaining defaults are robust across indoor/outdoor datasets [8].

OpenRatSLAM - Parameter descriptors

Table 3.1 summarizes the principal parameters and their roles as described by Ball *et al.*

Table 3.1: OpenRatSLAM — Parameter descriptions.

Parameter	**Description**
Visual odometry	
[vtrans_image_x_min, vtrans_image_y_min], [vtrans_image_x_max, vtrans_image_y_max]	These four parameters allow the specification of the cropping region for translational velocity.
[vrot_image_x_min, vrot_image_y_min], [vrot_image_x_max, vrot_image_y_max]	These four parameters allow the specification of the cropping region for rotational velocity.
camera_fov_deg	The horizontal camera field of view which is used to scale the rotational velocity.
camera_hz	The camera frame rate which is used to scale the velocities by accounting for the time between frames.
vtrans_scaling	This parameter directly scales the translation velocity into meters per second.
vtrans_max	This parameter limits the maximum translation velocity to handle large changes in illumination.
Local view parameters	
vt_panoramic	Set this to 1 if the images are panoramic.
vt_shift_match	The range (in pixel units) of horizontal offsets over which the current image is compared to all learnt image templates. Unused in panoramic mode.
vt_step_match	The number of pixels to increment the shift match offset.

Continues on next page

Table 3.1 (continued)

Parameter	**Description**
[image_crop_x_min, image_crop_y_min], [image_crop_x_max, image_crop_y_max]	These four parameters allow a cropping region of the original camera image to be specified. Cropping is a useful tool for specifying image regions that are salient for place localization. For example, carpet or road can be removed from the image. *Note these are defined from the top left of the image.*
[template_x_size, template_y_size]	The horizontal and vertical size in pixels of the 'sub-sampled' template that represents the camera view. For a single intensity profile set `template_y_size` to 1.
vt_match_threshold	The sensitivity parameter that determines the boundary between the current visual scene being considered novel and being matched to an already learnt visual template.
vt_normalisation	All templates are normalized by scaling their mean to this parameter (values clipped between 0 and 1), addressing global illumination changes.
vt_patch_normalisation	Increases local contrast of patch regions across the current view to handle local lighting changes and bring out more detail; the value sets the patch size (in pixels) from its centre.
Pose cell parameters	
pc_dim_xy, s_{xy}	The side length of the square (x, y) plane of the pose cell network. Larger networks increase computation but reduce the likelihood of hash collisions in the pose cell network and local view cells that can cause false loop closures in the experience map.
exp_delta_pc_threshold	The radius within the pose cell network that can be associated with a single experience—if the centroid of the pose cell activity packet moves more than this distance, a new experience is generated, regardless of whether the visual scene has changed.
pc_cell_x_size	A scaling factor to suit the platform's translational velocity range. The implementation limits packet movement to one cell per iteration to keep dynamics within the normal operating range.

Continues on next page

Table 3.1 (continued)

Parameter	Description
pc_vt_inject_energy	Determines the amount of energy injected into the pose cell network when a familiar scene is recognized. High values produce one-shot localization but increase brittleness to false matches; low values are robust but may require long sequences of familiar input to close loops.
vt_active_decay	Local view saturation mechanism that rapidly attenuates activity under repeated exposure to the same scene, reducing false loop closures while stationary (higher values shorten the effective injection period).
pc_vt_restore	Rate at which a local view cell is restored after being attenuated by repeated activations.
Experience map parameters	
exp_loops	The number of complete experience map graph relaxation cycles to perform per system iteration.

Start from defaults; adjust `vt_match_threshold` to balance novelty vs. matching, then (if needed) `pc_vt_inject_energy` to trade one-shot relocalization against false positives; PC dynamics are otherwise rarely altered [8].

While OpenRatSLAM inherits RatSLAM's resilience to odometric slip and perceptual ambiguity, its visual front-end remains template- and intensity-based, rendering it susceptible to severe appearance change, turbidity, and low-texture scenes. These implementation-level constraints motivate the cortically inspired enhancements introduced later in this dissertation (NeoSLAM), particularly for challenging underwater environments where ORB-SLAM3 collapses and conventional appearance cues are unreliable [8].

3.2 DolphinSLAM

While RatSLAM provided a robust 2D model for terrestrial navigation, its direct application to the volumetric and perceptually challenging underwater domain is limited. Addressing this, SILVEIRA [53] introduced **DolphinSLAM**, a biologically-inspired framework that extends the core hippocampal principles of RatSLAM to a full 3D underwater environment. The system was designed to handle multimodal sensory inputs (camera and sonar) and to be robust to the significant appearance changes caused by turbidity and variable illumination. The primary contributions of DolphinSLAM include its linear computational complexity with respect to the number of stored landmarks, its integration of a probabilistic place recognition front-end, and its open-source availability for future research [53].

3.2.1 System Overview

The DolphinSLAM architecture follows a modular structure, separating perception, motion estimation, and cognitive mapping, as shown in Figure 3.6. It operates using two coordinate systems (Figure 3.7): a global reference frame (**G**), typically aligned with the North-East-Down (NED) convention, and a local robot frame (**R**) fixed to the AUV's body, with the x-axis pointing forward, the y-axis to starboard, and the z-axis downward.

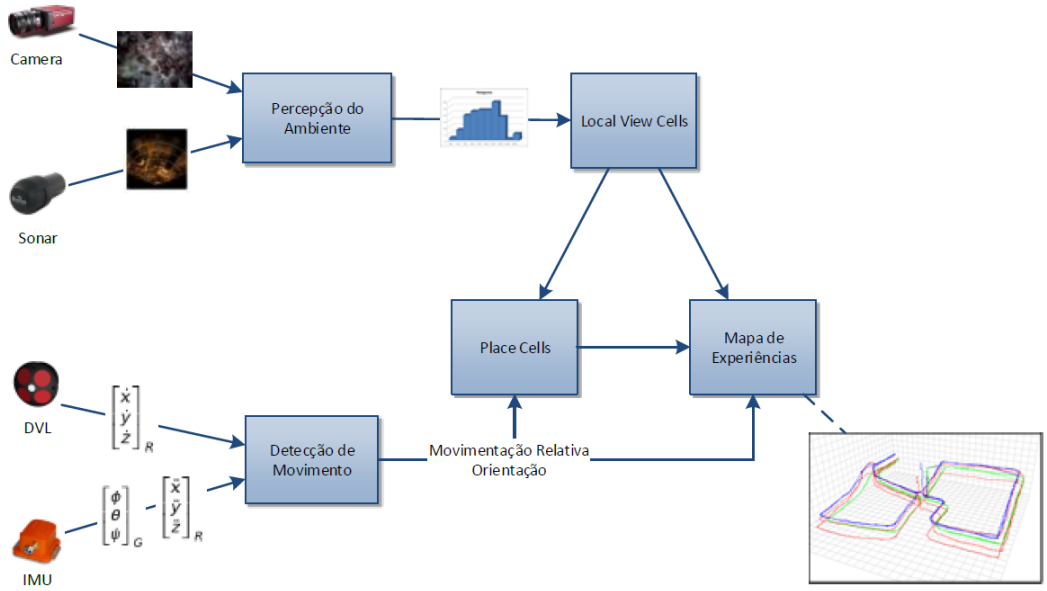


Figure 3.6: The modular architecture of the DolphinSLAM framework. Source: Reproduced from [53].

3.2.2 Front-End

Unlike the template-matching front-end of RatSLAM, DolphinSLAM integrates a more robust probabilistic visual place recognition system based on the FAB-MAP algorithm. This approach is better suited to handle the severe appearance variations common in underwater imagery.

Feature Extraction and Bag-of-Words Representation

The front-end processes both optical and acoustic images. For visual data, it extracts 128-dimensional SURF features [54] from Hessian keypoints. For acoustic data from imaging sonars, it uses the seven Hu moments [55], which are invariant to translation, rotation, and scale.

These low-level features are then clustered using k-means [56] to create a vocabulary of "visual words". Each new image is subsequently represented as a **Bag-of-Words (BoW)**

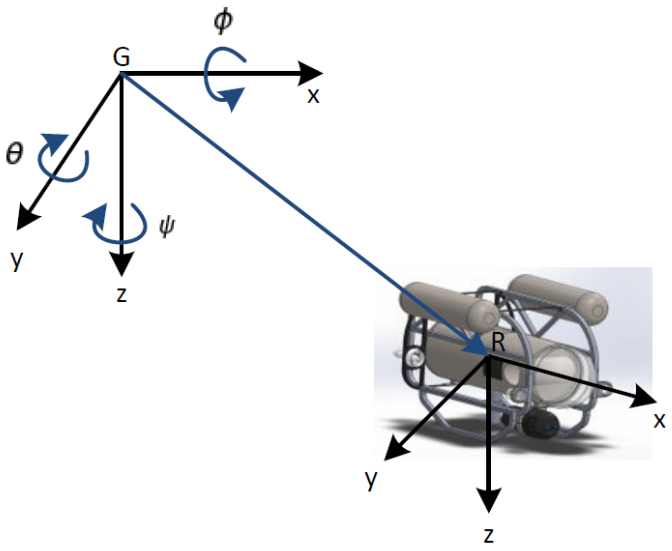


Figure 3.7: The global (G) and robot (R) coordinate systems used in DolphinSLAM. Source: Reproduced from [53].

histogram, which counts the frequency of each visual word in the image, as illustrated in Figure 3.8. This representation provides a degree of invariance to viewpoint changes.

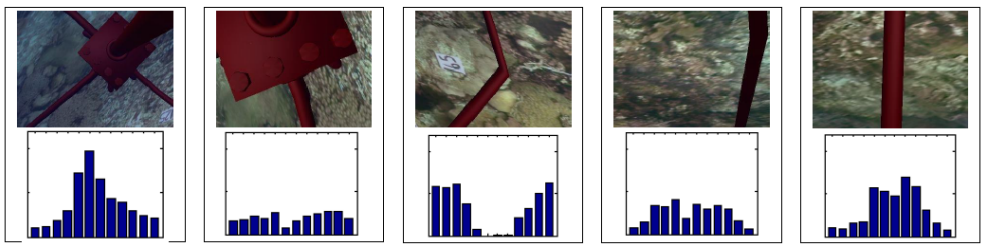


Figure 3.8: Example of Bag-of-Words histograms used in DolphinSLAM's front-end to represent visual scenes. Source: Reproduced from [53].

Local View Cells and Probabilistic Matching

The FAB-MAP algorithm uses the BoW representation to estimate the probability that a new image corresponds to a previously visited location (p_i) versus a new, unseen location (p_{new}). When the probability of a new location exceeds a predefined threshold, a new **Local View Cell (LVC)** is created with an activation rate $l_{new} = 1.0$ (Figure 3.10). The set of all LVCs, L, is defined as:

$$L = \{l_1, \ldots, l_i, \ldots, l_m\} \tag{3.8}$$

where m is the total number of LVCs in the system.

If a previously seen location is recognized, the corresponding LVC is activated with a rate proportional to its match probability, p_i (Figure 3.11), if it exceeds a sensitivity threshold P_s:

$$l_i = p_i \Leftrightarrow p_i \geq P_s \tag{3.9}$$

Otherwise, $l_i = 0$. The set of active LVCs provides the external sensory input to the Pose Cell Network.

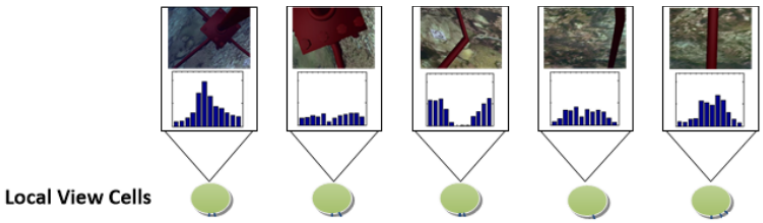


Figure 3.9: Conceptual representation of Local View Cells. Source: Reproduced from [53].

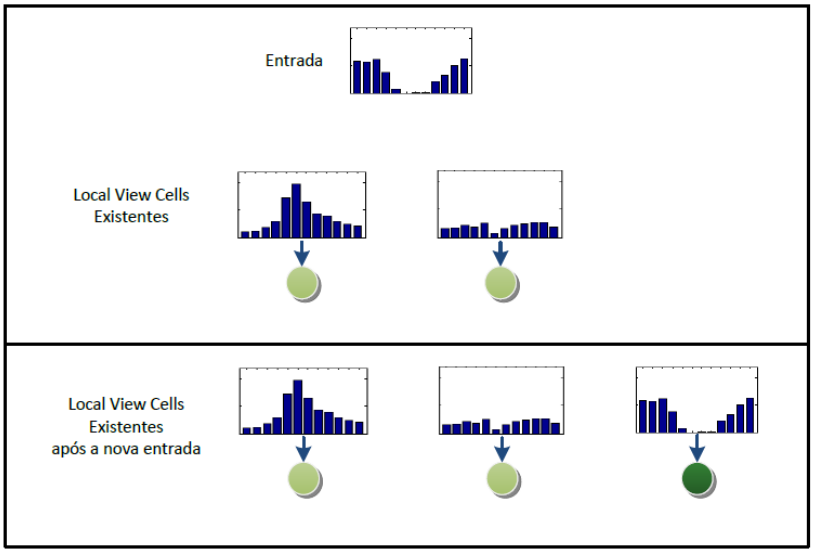


Figure 3.10: Creation of a new Local View Cell. Source: Reproduced from [53].

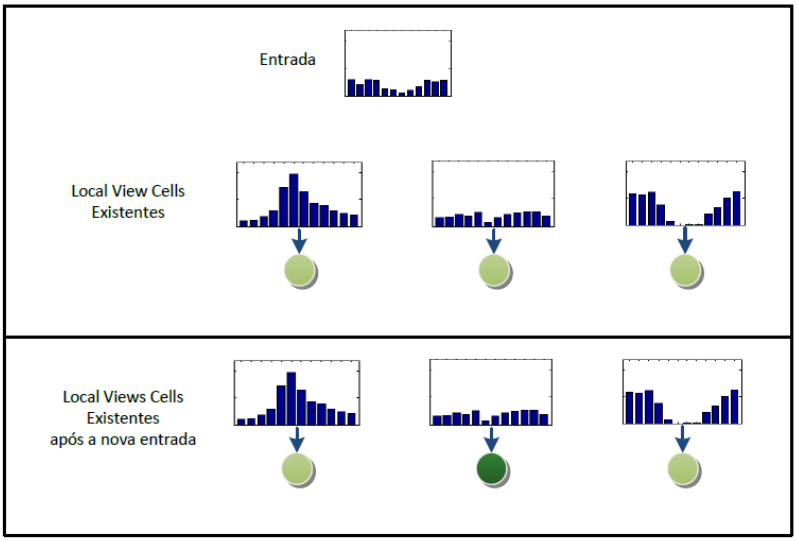


Figure 3.11: Activation of an existing Local View Cell. Source: Reproduced from [53].

3.2.3 Back-End

The core innovation of DolphinSLAM is its extension of the RatSLAM back-end to three dimensions, enabling true volumetric mapping and localization.

3D Path Integration

The motion estimation module fuses data from a Doppler Velocity Log (DVL) and an Inertial Measurement Unit (IMU) to compute the AUV's velocity. This velocity is transformed from the robot frame (**R**) to the global frame (**G**) using the standard Euler angle rotation matrix $R(\phi, \theta, \psi)$:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}^{G} = R(\phi, \theta, \psi) \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}^{R} \tag{3.10}$$

The incremental displacement over a time interval Δ_t is then calculated and integrated to update the vehicle's position:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}^{G}_{t+1} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}^{G}_{t} + \Delta t \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}^{G}_{t} \tag{3.11}$$

This odometric information is used for path integration within the Pose Cell Network and for creating edges in the Experience Map.

3D Pose Cell Network

DolphinSLAM implements a 3D Continuous Attractor Neural Network (CANN) of place cells [57, 58], inspired by evidence of 3D place cells in bats [59]. The network represents the AUV's pose belief as a stable "packet of activation" within a 3D lattice of neurons (Figure 3.12). The dynamics of the network are governed by four sequential steps:

1. **Lateral Excitation:** Recurrent connections between neurons update the network's activity. The synaptic weight ϵ is modeled using a Mexican-hat function to enforce local excitation and distal inhibition:

$$\epsilon = \left(1 - \frac{d^2}{\sigma^2}\right) e^{\frac{-d^2}{2\sigma^2}} \tag{3.12}$$

where d is the toroidal distance between two neurons in the lattice, calculated as:

$$d = \sqrt{d_{x'}^2 + d_{y'}^2 + d_{z'}^2} \tag{3.13}$$

with the wrapped distance on each axis given by:

$$\begin{aligned} d_{x'} &= \min\left(\left|x' - a\right|, n_{x'} - \left|x' - a\right|\right) \\ d_{y'} &= \min\left(\left|y' - b\right|, n_{y'} - \left|y' - b\right|\right) \\ d_{z'} &= \min\left(\left|z' - c\right|, n_{z'} - \left|z' - c\right|\right) \end{aligned}$$

The total excitation for a neuron $r_{x'y'z'}$ is the convolution of the network's current activity with this kernel:

$$r_{x'y'z'} = \sum_{a=1}^{n_{x'}} \sum_{b=1}^{n_{y'}} \sum_{c=1}^{n_{z'}} \varepsilon r_{abc} \tag{3.14}$$

2. **Path Integration:** The activity packet is shifted within the 3D lattice according to the displacement computed from the odometry, as illustrated conceptually in Figure 3.13.

3. **External Input Injection:** The activations of the LVCs provide a corrective sensory signal (Figure 3.14). The synaptic strength β between an LVC l_i and a pose cell $r_{x'y'z'}$ is learned via a Hebbian rule, reinforcing connections when both cells are simultaneously active:

$$\beta_{ix'y'z'} = \max\left(\beta_{ix'y'z'}, \lambda\, l_i\, r_{x'y'z'}\right) \tag{3.15}$$

where λ is the learning rate. The total external excitation received by a neuron is:

$$\Delta r_{x'y'z'} = \sum_{i=1}^{m} \beta_{ix'y'z'}\, l_i \tag{3.16}$$

4. **Normalization:** The total activity in the network is normalized to prevent un-

bounded growth and maintain a stable activation packet. The normalization factor η is the maximum activity in the network:

$$\eta = \max(r_{x'y'z'}) \tag{3.17}$$

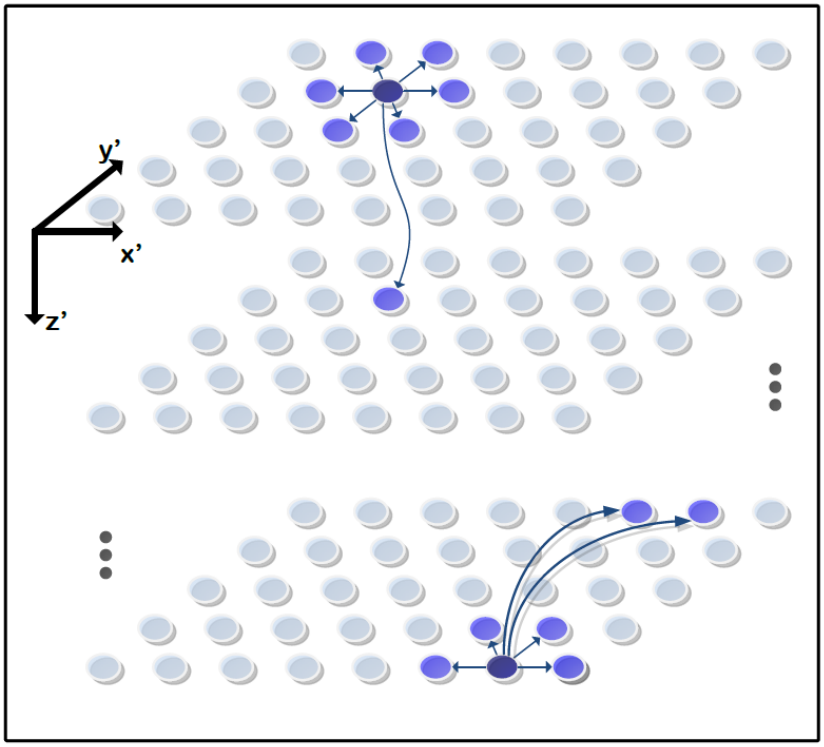


Figure 3.12: A 3D Continuous Attractor Neural Network showing a stable "packet of activation" that represents the system's pose belief. Source: Reproduced from [53].

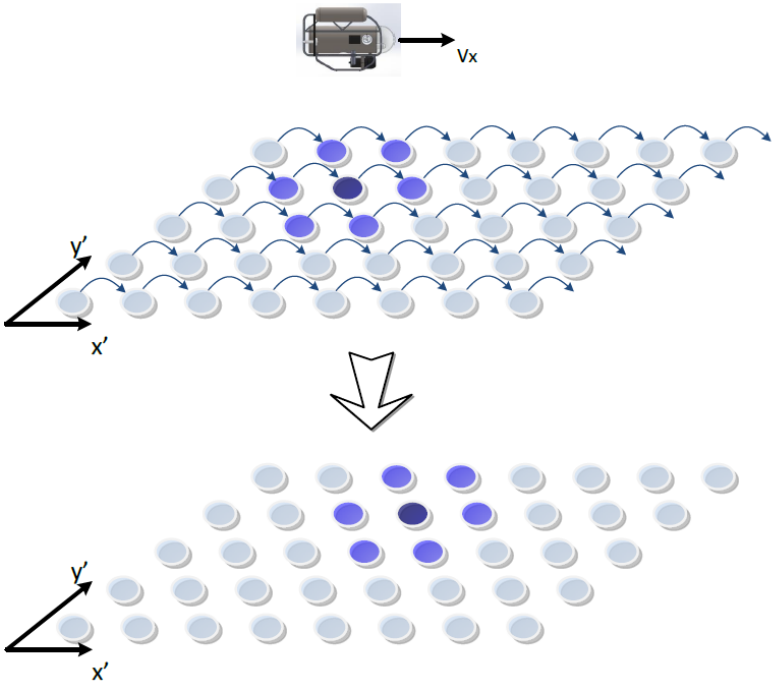


Figure 3.13: Conceptual model of path integration in the Pose Cell Network. Source: Reproduced from [53].

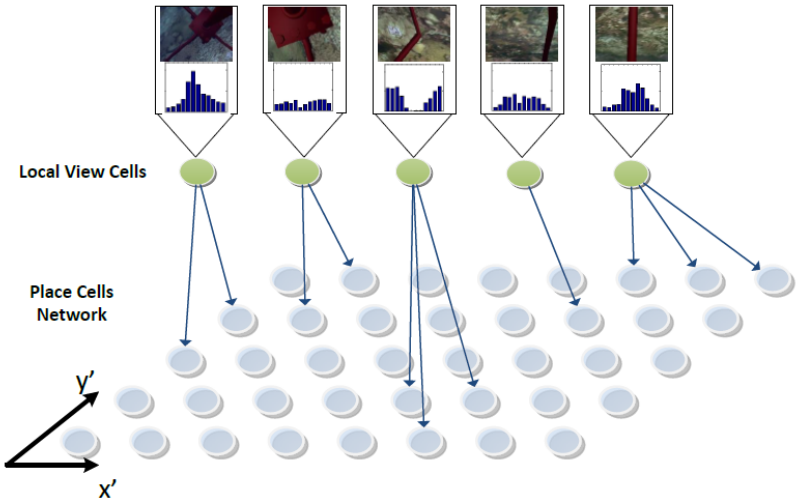


Figure 3.14: Connection between Local View Cells and Place Cells for external input injection. Source: Reproduced from [53].

3D Experience Map

The Experience Map is a semi-metric, topological graph that stores a globally consistent representation of the environment. Each node in the map, or "experience" $e_i = \{R^i, L^i, p_i\}$, binds the state of the Pose Cell Network (R^i) and the active LVCs (L^i) to a 3D spatial coordinate (p_i).

A new experience is created when the current neural state is sufficiently dissimilar to

all existing experiences in the map, based on a similarity metric S^i:

$$S^i = \alpha_R \left| R^i - R \right| + \alpha_L \left| L^i - L \right| \tag{3.18}$$

where α_R and α_L are weighting constants. If $\min(S^i)$ is above a threshold, a new experience e_j is created at a position relative to the previous experience e_i and the odometric displacement Δp^{ij}:

$$e_j = \{R^j, L^j, p^i + \Delta p^{ij}\} \tag{3.19}$$

A new topological link t_{ij} storing the odometry is also created:

$$t_{ij} = \{\Delta p^{ij}\} \tag{3.20}$$

When a previously visited location is recognized (a loop closure), the system creates a new experience e_j and two new links: one from the previous experience (t_{ij}) and one to the recognized experience e_k (t_{jk}), with the latter's displacement Δp^{im} derived from visual information.

$$e_j = \{R^j, L^j, p^i + \Delta p^{ij}\} \tag{3.21}$$

$$t_{ij} = \{\Delta p^{ij}\} \tag{3.22}$$

$$t_{jk} = \{\Delta p^{im}\} \tag{3.23}$$

This event triggers a graph relaxation process. The accumulated odometric error ε_p is calculated as:

$$\varepsilon_p = p_j + \Delta p^{im} - p_k \tag{3.24}$$

This error is then uniformly distributed among the n links between the current and the recognized experience by updating each link with a correction Δt:

$$\Delta t = -\frac{\varepsilon_p}{n+1} \tag{3.25}$$

Finally, the 3D positions of all experiences are recalculated based on the corrected links, ensuring the global consistency of the map.

3.3 NeoSLAM

The Neocortex is the outer layer of the brain and is responsible for higher-order cognitive processes, such as perception, language, and conscious thought. It processes sensory information to form complex representations of the world. In contrast, the hippocampus is a seahorse-shaped structure primarily associated with memory formation and spatial navigation. In primates, the hippocampus receives sensory input through the entorhinal cortex, which integrates streams from multiple cortical association areas, including the visual cortex [9], [60].

PIZZINO *et al.* [9] presents *Neocortex-based Simultaneous Localization and Mapping* (NeoSLAM), a novel long-term Visual SLAM capable of providing robots with skills in order to perform SLAM in real time by recognizing places it has previously visited under variations in appearance and illumination. As RatSLAM relies on low-resolution visual templates and intensity-based matching mechanisms, it remains sensitive to strong appearance variations, illumination changes, and perceptual aliasing, NeoSLAM is proposed to deal with this problem [10].

Based on theories and models of neuroscience, NeoSLAM integrates computational models inspired by the neocortex and hippocampal-entorhinal regions. Considerable evidence indicates the neocortex and the hippocampus are interconnected brain regions that work together to support learning and memory processes. The neocortex processes sensory information and sends it to the hippocampus for consolidation and initial storage. Over time, memories are transferred to the neocortex for long-term storage. The hippocampus is crucial for the formation of episodic memories, while the neocortex provides a more stable repository for long-term memory storage. Their dynamic interaction enables us to acquire, store, and retrieve memories, forming the foundation of primate cognitive abilities [10].

As illustrated in Figure 3.15, the NeoSLAM architecture is divided into a perception-focused Front-end and a navigation-focused Back-end. The Front-end comprises the Deep Learning-based Encoder, the Neocortex Network (HTM), the Spatial-View Cell Network, and the Loop Closure Detector. The Back-end consists of the Pose Cell Network and the Experience Map. This modularity ensures that visual disturbances are filtered at the perceptual level before affecting the robot's pose belief [10].

At a high level, the Encoder extracts mid-level visual descriptors from a pretrained CNN and converts them into a compact binary space that is amenable to HTM processing. The Neocortex module learns spatial patterns and short temporal sequences (via *Spatial Pooler* and *Temporal Memory*), producing SDRs that capture context. The Spatial-View Cell Network module performs visual place recognition by combining a small sequence of SDRs with a union operator, thereby stabilizing recognition despite local noise or small view changes. These Spatial-View Cells matches feed a 3D CAN of Pose Cells over (x, y, θ), and the Experience Map integrates Spatial-View Cells and pose states in a topological graph that accumulates loop-closures over time [9].

3.3.1 Front-End

The front-end of NeoSLAM is responsible for converting the continuous stream of raw pixel data into a discrete set of robust, noise-tolerant place representations. This system uses the framework of the HTM Model [61], observing the structural and algorithmic properties of the Neocortex, and processes the input data in the form of SDRs. The coding process in the form of SDRs mirrors, at an abstract computational level, key principles observed in biological sensory processing. An advantage over these sparse neuronal activities is the efficiency of representing information, that is, by activating only a subset of neurons, the brain is able to achieve high capacity representational while minimizing energy consumption and

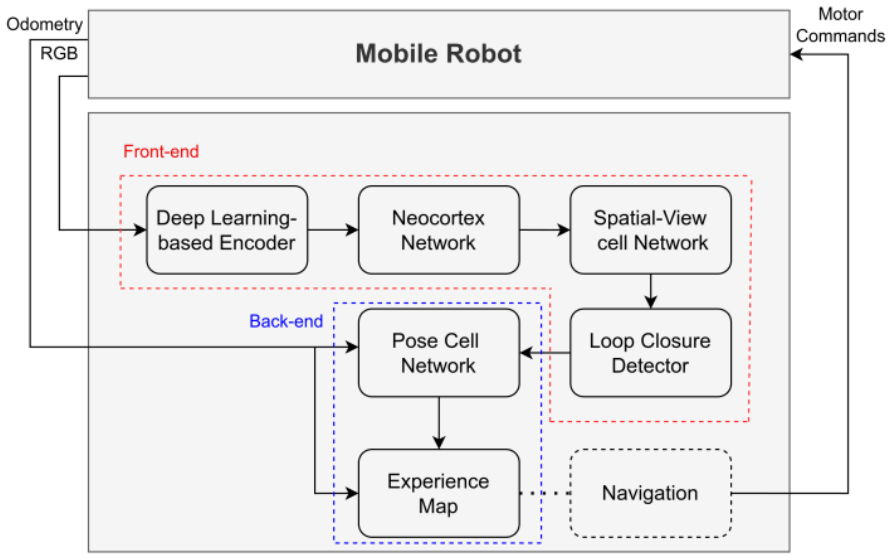


Figure 3.15: NeoSLAM System: Deep Learning-based Encoder converts raw input data into a format suitable for processing by the system. The Neocortex Network is responsible for creating a new descriptors based on the sequence processing capabilities of the neuronal model and its sparse distributed representations. Spatial-View Cell Network works such a way that these cells fire whenever the robot views a certain part of the environment, and is responsible for resetting the accumulative errors of the odometry. The Pose Cell Network represents the belief about the current pose. Finally, the Experience Map is a topological representation. Source: Reproduced from [10].

neuronal resources. This sparsity also allows the precise selection of coding of relevant features, resulting in the brain being efficient in extracting and processing information that stands out in this environment [9].

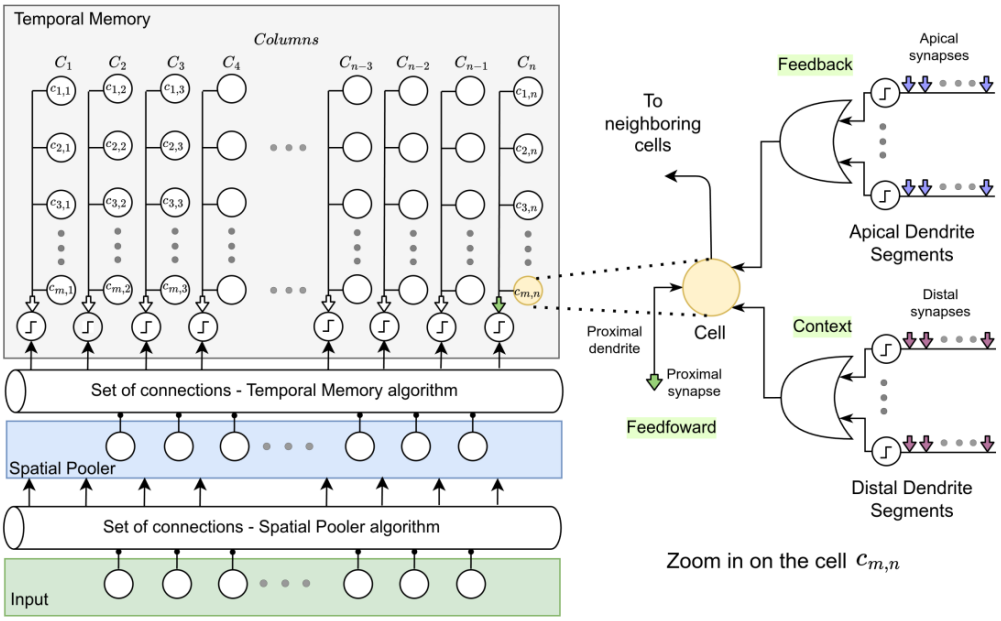


Figure 3.16: NeoSLAM: Hierarchical Temporal Memory Framework. Source: Reproduced from [10].

As shown in Figure 3.16, in a column, the cells have a shared proximal dendrite that connects to the input space (*Feedforward*) through a series of synapses, represented by a group of arrows. Each cell is illustrated with its distal (*Context*) and apical (*feedback*) dendrites, as shown on the right. Each dendrite segment contains multiple synaptic connections to other cells.

Thus, given a population of n_e neurons, an SDR can be represented in the form of an n-dimensional, binary, and sparse vector, that is:

$$\nu = [b_0, \ldots, b_{n_e-1}] \tag{3.26}$$

where $\nu \in \mathbb{S} : \{1, 0\}^{n_e}$ and only a small number of the units in this set are 1 (ones), thus defined as $\omega_\nu = \|\nu\|_1$.

The sparsity ς can be defined by ratio of activated neurons to total neurons, i.e,

$$\varsigma = \frac{\omega_\nu}{n_e} \tag{3.27}$$

and the capacity c_ν of SDR is defined by the volume of information in which SDR can embed,

$$c_\nu = \binom{n_e}{\omega_\nu} = \frac{n_e!}{\omega_\nu!\,(n_e - \omega_\nu)!} \tag{3.28}$$

The similarity between two SDR vectors is determined by an *overlap score*, i.e., the number of bits with value 1 that are coincident with both vectors. Let $\Phi : \mathbb{S} \times \mathbb{S} \to \mathbb{N}$ be the function defined by:

$$\Phi = \nu_1 \cdot \nu_2 \,, \tag{3.29}$$

where $\nu_1, \nu_2 \in \mathbb{S}^n$. A match between two SDR vectors occurs if the *overlap score* exceeds a threshold β, that is:

$$\Psi = \Phi\left(\nu_1, \nu_2\right) \geq \beta \,. \tag{3.30}$$

Therefore, β is typically configured such that

$$\beta \leq \min\left\{\omega_{\nu_1}, \omega_{\nu_2}\right\} \,. \tag{3.31}$$

Unlike the Hamming distance used in ORB-SLAM3 for binary descriptor matching, which penalizes any bitwise disagreement uniformly, similarity between Sparse Distributed Representations in NeoSLAM is based on overlap cardinality. Because SDRs are extremely sparse, a small amount of noise or bit corruption has a limited effect on the overlap score, yielding higher robustness to perceptual variations and sensor noise. This property is fundamental to the long-term place recognition capabilities of NeoSLAM.

Deep Learning-based Encoder

Convolutional Neural Networks have become fundamental tools in computer vision, with architectures inspired by early findings in the primary visual cortex [10]. These networks automatically extract hierarchical spatial features without the need for manual feature engineering, making them highly adaptable [10]. In the NeoSLAM framework, the encoder is responsible for transforming raw, high-dimensional visual data into Sparse Distributed Representations (SDRs) suitable for processing by the Neocortex Network module [9, 10].

PIZZINO [10] stated that the encoder is presented as a pipeline composed of three modules, shown in Figure 3.17: (i) a pre-trained CNN used as a feature extractor, (ii) dimensionality reduction (feature compression), and (iii) feature binarization, which converts continuous-valued features into a binary descriptor.

The choice of the CNN layer used for place recognition is based on the analysis of ConvNet features for place recognition, features extracted from *middle layers* exhibit robustness to appearance changes (e.g., time of day, seasons, weather), while features extracted from *top layers* are more robust to viewpoint changes [10, 62]. A practical constraint is that CNNs trained in a supervised manner typically require large labeled datasets to generalize well; therefore, they cannot be simply re-trained with a small number of images at deployment time, since that may lead to overfitting [10]. Therefore, three CNN-based strategies are adopted to build an encoder for neocortex models: (i) CNNs trained for other tasks (e.g., AlexNet pre-trained on ImageNet), (ii) CNNs designed for place recognition and long-term perceptual robustness (e.g., HybridNet pre-trained on SPED, with weights initialized from a pre-trained CaffeNet), and (iii) feature map pruning, in which less effective feature maps are removed [10].

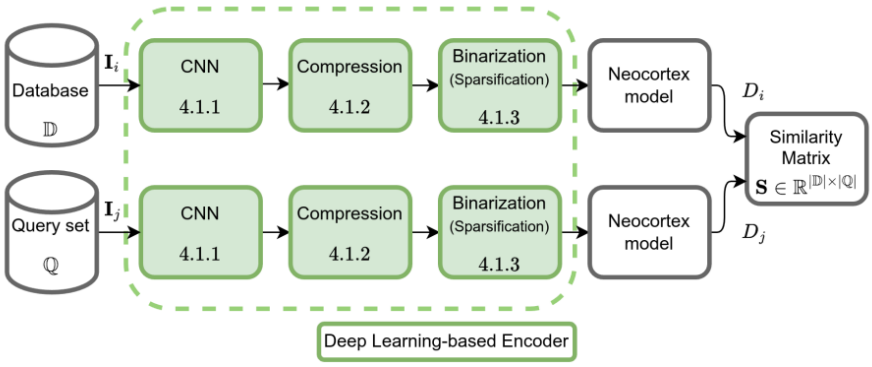


Figure 3.17: Deep Learning-based Encoder pipeline: Feature extraction (AlexNet-conv3), Gaussian Random Projection, and Binarization. Source: Reproduced from [10].

After feature extraction, the high-dimensional descriptor is reduced in dimension through random projections [63], which do not require training data and are robust to unexpected condition changes [10]. This step is supported by the Johnson–Lindenstrauss lemma [64], which guarantees that the pairwise distances between points in the original high-dimensional space and their projections into the lower-dimensional space will be approximately preserved. In [65], Gaussian Random Projection (GRP) [66] is established technique for dimensionality reduction of image descriptors in VPR applications.

In practice, GRP is described as a two-step procedure: First, a random projection matrix $P = [p_{ij}] \in \mathbb{R}^{L\times M}$ is generated with dimensions corresponding to the desired lower-dimensional space. The elements p_{ij} (for each column P_j of P) are drawn from a Gaussian distribution, $p_{ij} \sim \mathcal{N}(\mu = 0, \sigma = 1)$, and each column $P_j \in P$ is normalized so that $\|P_j\| = 1$ [10]. Second, each high-dimensional descriptor is projected by multiplying it with the random matrix P, resulting in a lower-dimensional representation of the original data [10].

Therefore, random projections can be used as descriptor compression for computational and memory efficiency [67]. With $P \in \mathbb{R}^{L\times M}$, it is possible a compression rate $\frac{L}{M}$, which maps M-dimensional descriptors to L-dimensions [10].

Binarization is then applied to obtain a binary descriptor compatible with HTM processing. Let $d \in \mathbb{R}^n$ be a real-valued descriptor and $d_b \in \mathbb{B}^n$ be its binary version; binarization is defined as a mapping

$$d \in \mathbb{R}^n \xrightarrow{b(\cdot)} d^b \in \mathbb{B}^n \tag{3.32}$$

PIZZINO [10] highlights three practical benefits: (i) compatibility with HTM networks, (ii) efficient similarity measurement via Hamming distance (as an XOR operator), and (iii) reduced memory usage .

Two binarization techniques are presented. The first is *Sign Random Projection* [68], which starts with a random projection

$$\bar{d} = P \cdot d, \qquad \bar{d} \in \mathbb{R}^L,\ d \in \mathbb{R}^M, \tag{3.33}$$

followed by mapping the sign of each component to a binary value:

$$d_j^b = \begin{cases} 1, & \text{if } \bar{d}_j > 0, \\ 0, & \text{if } \bar{d}_j \leq 0. \end{cases} \tag{3.34}$$

The second is *Sparse Locality Sensitive Binary Hashing (sLSBH)* [67], which ensures constant density in the binarized descriptor and allows defining a desired density. The binarized descriptor is formed by concatenating two binary vectors:

$$d^b = \left[d^b(+) \ ++ \ d^b(-) \right]^T, \tag{3.35}$$

where the vectors $d^b(+)$ and $d^b(-)$ are concatenated $(++)$ and defined as

$$d_j^b(+) = \begin{cases} 1, & \text{if } \bar{d}_j \text{ is amongst the } s \cdot \|\bar{d}\| \text{ largest values in } \bar{d}, \\ 0, & \text{otherwise,} \end{cases} \tag{3.36}$$

$$d_j^b(-) = \begin{cases} 1, & \text{if } \bar{d}_j \text{ is amongst the } s \cdot \|\bar{d}\| \text{ smallest values in } \bar{d}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.37}$$

where s is a user-defined parameter that controls the density in a binarized descriptor d^b [67]. PIZZINO [10] also notes that this mapping doubles the dimensionality; to preserve dimensionality, alternatives are discussed, including using only the largest values, only the smallest values, or the logical disjunction of half-density subsets.

Neocortex Network

The HTM algorithm mimics the Neocortex's ability to learn and recognize patterns using a hierarchical structure of neurons that process standardized time series. This enables the Neocortex to detect complex patterns over time and make predictions based on previous experiences. A key advantage of the HTM algorithm is its capacity to handle noisy or incomplete data. Its hierarchical structure allows it to recognize patterns even when the input is flawed or incomplete, enabling it to adapt to changing inputs and learn new patterns without requiring updates.

It can be seen in Figure 3.16 that n is the representation of the number of minicolumns (C_j, such that $j = 1, ..., n$), and m is the number of cells per column, we obtain a total of nm cells in the layer. The HTM Cells ($c_{i,j}$), then stacked in columns, form groups of HTM regions. In this way, cells within these regions gather information from three distinct sources, namely: Feed-forward, Context, and Feedback. These cells are in the *inactive*, *active* or *predicted* states such that links between cells comprise a collection of synapses. In the HTM Model, these synapses allow the sharing of information (displayed in binary values) between cells.

The Spatial Pooler (SP) algorithm processes input through Feed-forward input, while Temporal Memory (TM) provides Context input. The SP's main role in HTM Theory is to identify spatio-temporal patterns in the input data and convert them into SDRs continuously. In visual localization, the SP functions as a feature detector, extracting distinguishing properties from a region to help recognize a place. The SP's output reflects the activation of minicolumns in response to Feed-forward inputs. Meanwhile, the TM learns sequences by forming representations of previous Context inputs, determining which cells in the columns are active and learning the permanence of distal synapses. Learning occurs through a Hebbian-like rule, where synapses only connect cells with permanence values above a set threshold. TM's output represents the activation of individual cells across all minicolumns [69].

Spatial-View Cells Network

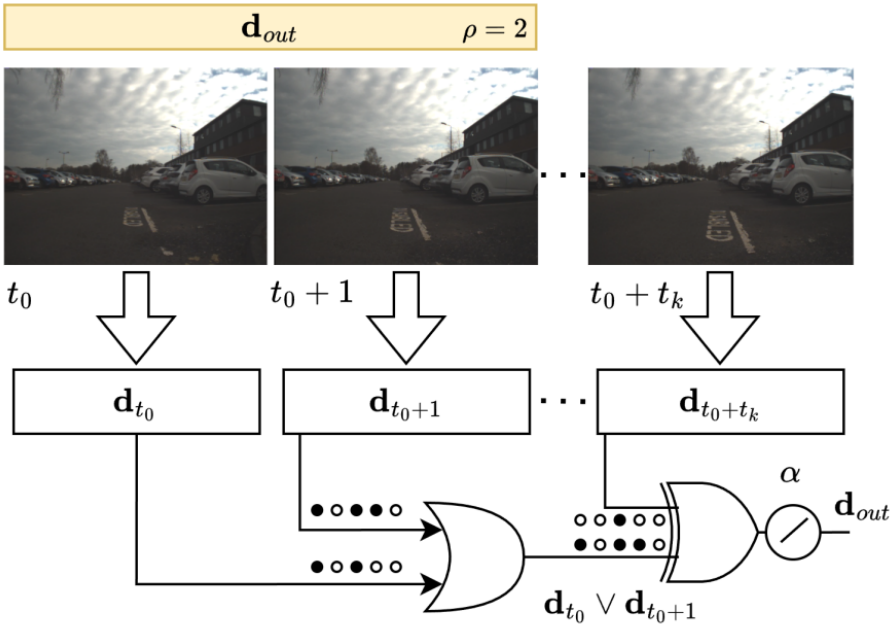


Figure 3.18: Spatial-View Cells. Source: Reproduced from [10].

This module activates Spatial-View Cells every time the robot observes a certain region of the environment. These cells indirectly correct accumulated odometric drift by injecting consistent place evidence into the Pose Cell network. Similar to RatSLAM [46, 47], these cells represent what the robot perceives. However, when a new visual scene is viewed, a new View Cell is not necessarily created and associated with the HTM descriptor. In this way, the features extracted from the images by the Module Encoder are represented in the form of SDRs, so that different descriptors have similar attributes according to active bits ($1's$) in the same place. Therefore, the advantage is to perform the OR operation on a set of SDRs, so that the resulting SDR preserves the attributes of the set and can be compared to determine membership in the set used to form the union [9, 10].

As described in Figure 3.18, a spatial vision cell model is inspired by neurons in the primate hippocampus that are activated when a specific area of the environment appears within the animal's field of view and utilizes the properties of sparse distributed representations.

Thus, given a set D that represents a sequence of SDR descriptors, defined as:

$$D = [\mathbf{d_{t_0}}, \ldots, \mathbf{d_{t_0+t_k}}] \tag{3.38}$$

where $\mathbf{d} \in \mathbb{S}^n$ represents images taken at a specific place and at particular instants in discrete time. The interval n_0 is defined by:

$$n_0 = [t_0, t_0 + t_k] = \{t_k \in \mathbb{N} : \Phi\left(\mathbf{d}_{t_k}, \mathbf{d}_{t_k+1}\right) \geq \alpha\} \tag{3.39}$$

The output of this model is the descriptor:

$$\mathbf{d}_{\text{out}} = \mathbf{d}_{t_0} \vee \cdots \vee \mathbf{d}_{t_0+t_k} \tag{3.40}$$

The parameter α can be understood as the maximum overlap in order to control the sparsity of the results. The maximum size of the interval of the Spatial-View Cells that encloses representations with similarity between SDRs is defined by the parameter ρ, so that:

$$n_0 = [0, \min(k, \rho)] \tag{3.41}$$

This method helps to avoid the corridor problem, i.e., the absence of significant structure along both outdoor and indoor environments [9].

3.3.2 Back-End

The back-end of NeoSLAM is responsible for spatial reasoning and is based on the robust, well-established architecture of RatSLAM.

Pose Cell Network

As in RatSLAM, pose is represented by a three-dimensional Continuous Attractor Neural Network over (x, y, θ). The activity within this lattice integrates motion cues from odometry (via *path integration*) and is corrected by place recognition matches from the Spatial-View Cells (via *activity injection*). This neural representation supports robustness to noise and smooths pose estimates between loop-closures [9].

Experience Map

The Experience Map fuses Spatial-View and Pose Cell states into a topological graph. Each node encodes a distinct Spatial-View Cells/pose state; when the state changes beyond existing nodes' tolerances, a new *experience* is created and linked to the previous one. Over

time this yields a sparse, loop-closed graph of the workspace that is updated online and optimized in the back-end [9].

3.3.3 Software Architecture

To guarantee repeatable results across heterogeneous machines, NeoSLAM is packaged inside a *Singularity* container tailored for scientific/HPC workflows. Containerization encapsulates the exact operating system, libraries, and run-time configuration required by the pipeline, yielding (i) portability across clusters and cloud platforms, (ii) strict software reproducibility upon updates or reinstallation, (iii) isolation from host conflicts, and (iv) simplified backup as a single image file. Notably, the provided image executes reliably across multiple Ubuntu releases and can be deployed without root privileges (*user-space* execution) [10].

The NeoSLAM image co-locates three core frameworks and their dependencies: *ROS Melodic* (the last ROS distribution compatible with Python 2.7), *Numenta NuPIC* (implementations of HTM algorithms used by the neocortical front-end), and *PyTorch/Torchvision* (CNN feature extraction, including AlexNet). This combination motivates containerization: NuPIC constrains Python to 2.7 and hence ROS to Melodic (Ubuntu 18.04), whereas Torch/Torchvision manage modern CNN inference; the container reconciles these stacks into a coherent, reproducible environment [10].

In the methods chapter, explicitly state that all experiments (including underwater evaluations) were executed from a frozen Singularity image bundling ROS Melodic, NuPIC (HTM), and PyTorch/Torchvision; emphasize the reproducibility and host-agnostic deployment as a design choice enabling fair comparisons to RatSLAM and ORB-SLAM baselines [10].

ROS Implementation

NeoSLAM ROS Architecture , shown in Figure 3.19,follows the standard ROS *distributed* model: functionality is decomposed into nodes that communicate via topics and are configured at run time using the parameter server. The neocortical module is implemented as a *ROS Action Server* in Python to handle long-running, goal-oriented computations (sequence learning and prediction), while the visual cortex node streams images to this server and consumes the published Spatial-View (Spatial-View Cells) responses. Back-end mapping and navigation (Pose Cells and Experience Map) are adapted from OpenRatSLAM in C++, with configuration exposed through ROS parameters [10].

The computation graph includes dedicated topics for (i) broadcasting the evolving topological map, (ii) exchanging Spatial-View Cells templates, and (iii) issuing topological actions from the Pose Cell network; the main topics illustrated in the ROS graph are: `/ExperienceMap/Map`, `/SpatialView/Template`, and `/PoseCell/TopologicalAction`. These support loose coupling between perception, loop-closure, and graph updates, and make the system readily interoperable with other ROS packages (e.g., logging, visualiza-

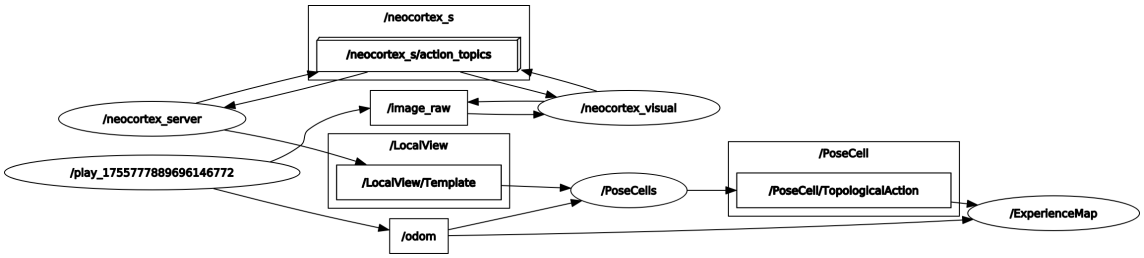


Figure 3.19: NeoSLAM ROS Graph for *Robotarium* dataset. Source: Author's own creation based on [10].

tion, or alternative odometry sources).

Add a concise ROS graph overview (with or without an `rqt_graph` figure) describing the Action Server interface of the Neocortex node and the roles of the three topics above. Clarify that the Python (HTM) and C++ (RatSLAM back-end) components are bridged through ROS topics and the parameter server for dataset-specific tuning [10].

Parameter Configurations

The container hosts *NuPIC* implementations of the Spatial Pooler (SP) and Temporal Memory (TM) used in NeoSLAM's neocortical front-end. Tables 3.2 and 3.3 consolidate the main data structures, routines, and hyperparameters, with recommended settings for typical visual place recognition (VPR) tasks, aligning BAMI's defaults with the ranges used in NeoSLAM experiments [9, 10, 70].

Table 3.2: Spatial Pooling: Data structures, routines, and parameters.

Item	**Description and recommended settings**
Data structures and variables	
`Columns`	List of all SP columns (mini-columns).
`columnCount`	Total number of columns in the region; task-dependent. **Recommended:** at least 2048.
`input(t, j)`	Input bit j at time t (binary).
`overlap(c)`	Overlap score of column c with the current input (sum of active connected synapses, optionally multiplied by `boost(c)`).
`activeColumns(t)`	Indices of columns that win inhibition at time t.
`inhibitionRadius`	Average connected receptive-field size across columns (computed internally).
`neighbors(c)`	Columns within `inhibitionRadius` of column c.
`synapse`	Structure with a permanence value and a source input index.

Continues on next page

Table 3.2 (continued)

Item	Description and recommended settings
`potentialSynapses(c)`	All potential synapses (and permanences) for column c.
`connectedSynapses(c)`	Subset of `potentialSynapses(c)` with permanence > `connectedPerm`.
`activeDutyCycle(c)`	Sliding average of how often c has been active after inhibition (e.g., last ∼1000 iterations).
`overlapDutyCycle(c)`	Sliding average of how often c has had overlap ≥ `stimulusThreshold` (e.g., last ∼1000 iterations).
Supporting routines (pseudocode)	
`kthScore(cols, k)`	Returns the k-th highest overlap value among `cols`.
`updateActiveDutyCycle(c)`	Updates the moving average for c's post-inhibition activity (used by boosting).
`updateOverlapDutyCycle(c)`	Updates the moving average for c's pre-inhibition overlap (used by boosting).
`boostFunction(·)`	Exponential function of duty cycles that sets `boost(c)`.
`increasePermanences(c, δ)`	Uniformly increases all permanences of c to aid learning when needed (*BAMI example:* $\delta = 0.1 \cdot$ `connectedPerm`).
`averageReceptiveFieldSize()`	Recomputes `inhibitionRadius`.
Core parameters and typical settings	
`columnDimensions`	Output topology (e.g., `[2048]` or `[64, 64]` for the same count). Choose per task and hardware.
`inputDimensions`	Input topology; must match dimensionality with `columnDimensions`.
`potentialRadius`	Radius in input space that a column can potentially connect to (small ⇒ local RF; large ⇒ near-global).
`globalInhibition`	If `True`, winners are selected globally (often used in practice for speed/robustness); otherwise inhibition is local.
`localAreaDensity`	Target fraction of active columns within an inhibition area (use either this *or* `numActiveColumnsPerInhArea`).
`numActiveColumnsPerInhArea`	Max winners per inhibition area. **Rule of thumb:** ∼2% of total columns under global inhibition; *e.g.* `40` for `2048`. **Minimum:** `25`.

Continues on next page

Table 3.2 (continued)

Item	Description and recommended settings
`stimulusThreshold`	Minimum active inputs required for a column to be considered during inhibition; typically `0`–`5`. If unsure, set `0`.
`potentialPct`	Fraction of inputs within the potential radius to initialize as potential synapses. Set so that at initialization each column has at least **15–20** connected inputs; *example:* if typical input has 40 ON bits and 50% of synapses start connected, use `potentialPct` ≥ 0.75 because $40 \times 0.5 \times 0.75 \approx 15$.
`connectedPerm`	Permanence threshold for a synapse to be considered connected; typically `0.2`.
`synPermActiveInc`	Permanence increment for active synapses during learning; typically a small value, e.g., `0.03`.
`synPermInactiveDec`	Permanence decrement for inactive synapses during learning; typically smaller than the increment, e.g., `0.015`.
`boostStrength`	≥ 0; controls boosting strength (`0` disables boosting).
`minPctOverlapDutyCycle`	Controls the minimum acceptable overlap duty cycle; below this, column permanences are uniformly boosted.

Table 3.3: Temporal Memory: Data structures, routines, and parameters.

Item	Description and recommended settings
Data structures and variables	
t	Discrete time step index for TM updates.
`columns`	List of all mini-columns in the layer.
`segments`	List of all distal dendrite segments across all cells. Each segment stores synapses (source cell indices and permanences).
`activeColumns`(t)	Set of active columns at time t (output of Spatial Pooler to TM).
`activeCells`(t)	Set of cells active at time t (union of correctly predicted cells and bursting cells).
`winnerCells`(t)	Subset of `activeCells` that drive learning at t (used to grow/update segments).

Continues on next page

Table 3.3 (continued)

Item	Description and recommended settings
`activeSegments`(t)	Segments with count of active *connected* synapses $\geq$ `ACTIVATION_THRESHOLD`.
`matchingSegments`(t)	Segments with count of active *potential* synapses $\geq$ `LEARNING_THRESHOLD` (eligible for growth).
`numActivePotentialSynapses` (t, *segment*)	Number of active potential synapses on a segment at time t; used to score matches and select growth targets.
Supporting routines (pseudocode)	
`segmentsForColumn` (*column*, *segments*)	Return all segments that belong to any cell in the given column.
`count`(*collection*)	Cardinality utility for sets/lists used in scoring and thresholds.
`leastUsedCell` (*column*)	Selects a learning cell with the fewest existing segments in the column (break ties randomly).
`bestMatchingSegment`(*column*)	Among `matchingSegments`, returns the segment with the largest number of active potential synapses.
`growSynapses` (*segment*, *newSynapseCount*)	Adds synapses from prior `winnerCells` to a segment (no duplicates), initializing with `INITIAL_PERMANENCE`.
Core parameters and typical settings	
`ACTIVATION_THRESHOLD`	Minimum number of active *connected* synapses for a segment to be *active*. **Typical:** `4` (robust to appearance change in VPR/NeoSLAM).
`LEARNING_THRESHOLD`	Minimum number of active *potential* synapses for a segment to be *matching*. **Typical:** $\leq$ `ACTIVATION_THRESHOLD` (e.g., `3-4`).
`CONNECTED_PERMANENCE`	Permanence threshold above which a synapse is considered connected. **Typical:** `0.2`.
`INITIAL_PERMANENCE`	Permanence assigned to newly created synapses. **Recommended:** near `CONNECTED_PERMANENCE`, slightly below (e.g., `0.18-0.20`) to require reinforcement.
`PERMANENCE_INCREMENT`	Additive increase for active synapses on correctly predictive segments. **Typical:** small, e.g., `0.03`.
`PERMANENCE_DECREMENT`	Additive decrease for inactive synapses on predictive segments. **Typical:** slightly smaller than increment, e.g., `0.015`.

Continues on next page

Table 3.3 (continued)

Item	Description and recommended settings
`PREDICTED_DECREMENT`	Penalty when a segment predicted but its cell did not become active. **Typical:** small positive (e.g., `0.001-0.01`); often disabled.
`SYNAPSE_SAMPLE_SIZE`	Target number of presynaptic connections grown on a learning segment (*sub-sampling* of the SDR). **Typical:** `15-20`.
`cellsPerColumn`	Number of TM cells per mini-column (controls Context capacity and sequence order). **Typical:** `32` for VPR tasks.

3.4 ORB-SLAM3

3.4.1 ORB Features

ORB-SLAM3 is built upon ORB features, from which the system derives its name. ORB stands for *Oriented FAST and Rotated BRIEF*, a feature detection and description method specifically designed to provide a favorable trade-off between computational efficiency, robustness, and invariance properties [17]. ORB features constitute the backbone of all data association processes in ORB-SLAM3 [4].

ORB combines two classical computer vision techniques: FAST for keypoint detection and BRIEF for feature description, extending both to achieve rotation invariance and improved robustness.

FAST (Features from Accelerated Segment Test) is a corner detection algorithm that identifies interest points by examining the intensity contrast between a candidate pixel and a circular set of surrounding pixels. FAST is computationally efficient and well suited for real-time applications; however, in its original formulation, it lacks orientation information and scale invariance [17].

In ORB, FAST is used as the initial keypoint detector, but its output is augmented by estimating a dominant orientation for each keypoint. This orientation is computed using the intensity centroid method, which assigns a consistent rotation to the detected feature based on local image moments. This extension enables rotation invariance, a critical requirement for SLAM systems operating under arbitrary camera motion [17].

BRIEF (Binary Robust Independent Elementary Features) is a binary descriptor that encodes local image appearance by performing a set of pairwise intensity comparisons within a smoothed image patch. The result is a compact binary string that allows extremely fast matching using the Hamming distance [17].

Despite its efficiency, the original BRIEF descriptor is not rotation invariant. ORB addresses this limitation by introducing a rotated version of BRIEF, where the sampling

pattern is steered according to the keypoint orientation estimated during the FAST stage. This modification preserves descriptor consistency under in-plane rotations [17].

By combining oriented FAST keypoints with rotated BRIEF descriptors, ORB achieves:

- rotation invariance,
- robustness to moderate viewpoint and illumination changes,
- low memory footprint,
- fast descriptor matching using Hamming distance.

These properties make ORB particularly suitable for real-time SLAM systems, where thousands of features must be detected and matched at high frame rates [4, 17].

Classical feature descriptors such as SIFT (Scale-Invariant Feature Transform) and SURF (Speeded-Up Robust Features) provide strong invariance to scale, rotation, and illumination changes and are highly discriminative. However, they rely on floating-point descriptors and complex operations, resulting in significantly higher computational cost [17].

In contrast, ORB features sacrifice some degree of scale invariance and descriptor distinctiveness in exchange for computational efficiency and real-time performance. This trade-off is well aligned with the requirements of ORB-SLAM3, where fast and reliable data association across frames, keyframes, and maps is more critical than maximal descriptor expressiveness [4].

In ORB-SLAM3, ORB features constitute the foundation of all data association mechanisms, including:

- short-term tracking between consecutive frames,
- mid-term matching between keyframes within the same map,
- long-term Place Recognition across previously visited areas,
- multi-map data association during map merging.

The use of binary ORB descriptors enables efficient matching via Hamming distance, which is essential for the scalability of the Atlas multi-map architecture and for maintaining real-time operation in large-scale environments [4].

3.4.2 System Overview

CAMPOS *et al.* [4] presents ORB-SLAM3, a feature-based SLAM system that supports monocular, stereo, and RGB-D cameras, employing both pin-hole and fisheye camera models. The system represents a significant evolution over previous ORB-SLAM versions by extending their capabilities to tightly-integrated visual-inertial estimation and seamless multi-map operation.

One of the main contributions of ORB-SLAM3 is a tightly-coupled Visual-Inertial SLAM formulation that consistently relies on Maximum-a-Posteriori (MAP) estimation, including during the IMU initialization phase. In contrast to loosely-coupled pipelines, inertial measurements are directly incorporated into the optimization problem alongside visual observations, enabling the system to estimate metric scale, gravity direction, velocities, and inertial sensor biases in a unified probabilistic framework. This formulation yields a robust real-time system capable of operating in small and large-scale environments, both indoors and outdoors, achieving significantly higher accuracy than previous visual-inertial approaches [4].

A second major contribution is the introduction of a multi-map architecture, denoted as the *Atlas*, combined with a high-recall Place Recognition strategy. This design allows the system to remain operational during prolonged periods of poor visual information. When tracking fails, ORB-SLAM3 initializes a new map rather than discarding accumulated information. Once previously mapped regions are revisited, disconnected maps are seamlessly merged. Unlike Visual Odometry systems, which rely exclusively on short-term observations, ORB-SLAM3 reuses information from all previously mapped areas across all stages of the algorithm, including tracking, local mapping, loop closing, and optimization.

A direct consequence of this long-term data reuse is the ability to include co-visible keyframes that may be widely separated in time or originate from different mapping sessions into the optimization process. These keyframes provide high-parallax observations, which are fundamental for accurate geometric reconstruction and drift correction.

In ORB-SLAM3, Bundle Adjustment (BA) constitutes the core optimization mechanism underlying both visual and visual-inertial estimation. BA is formulated as a MAP estimation problem in which camera (or body) states and 3D map points are jointly optimized by minimizing a cost function composed of measurement residuals weighted by their uncertainty.

In the pure visual case, BA minimizes the reprojection error of 3D landmarks onto image measurements. In the visual-inertial setting, this formulation is extended to incorporate inertial residuals derived from IMU preintegration. As a result, BA jointly optimizes camera poses, velocities, gravity direction, scale (when applicable), and inertial sensor biases. This tightly-coupled formulation allows inertial constraints to regularize motion estimation, particularly in low-texture or high-dynamic scenarios.

ORB-SLAM3 employs BA at multiple levels: local BA for real-time map refinement, welding BA during loop closing and map merging, and full BA for global refinement when computationally feasible. The selective activation of these optimizations balances accuracy and real-time performance.

Figure 3.20 shows the main system components:

1. **Atlas**: Is a Multi-map representation composed of a set of disconnected maps. There is an Active Map where the Tracking thread localizes the incoming frames, and is continuously optimized and grown with new keyframes by the Local Mapping thread. The system builds a unique DBoW2 database of keyframes that is used for

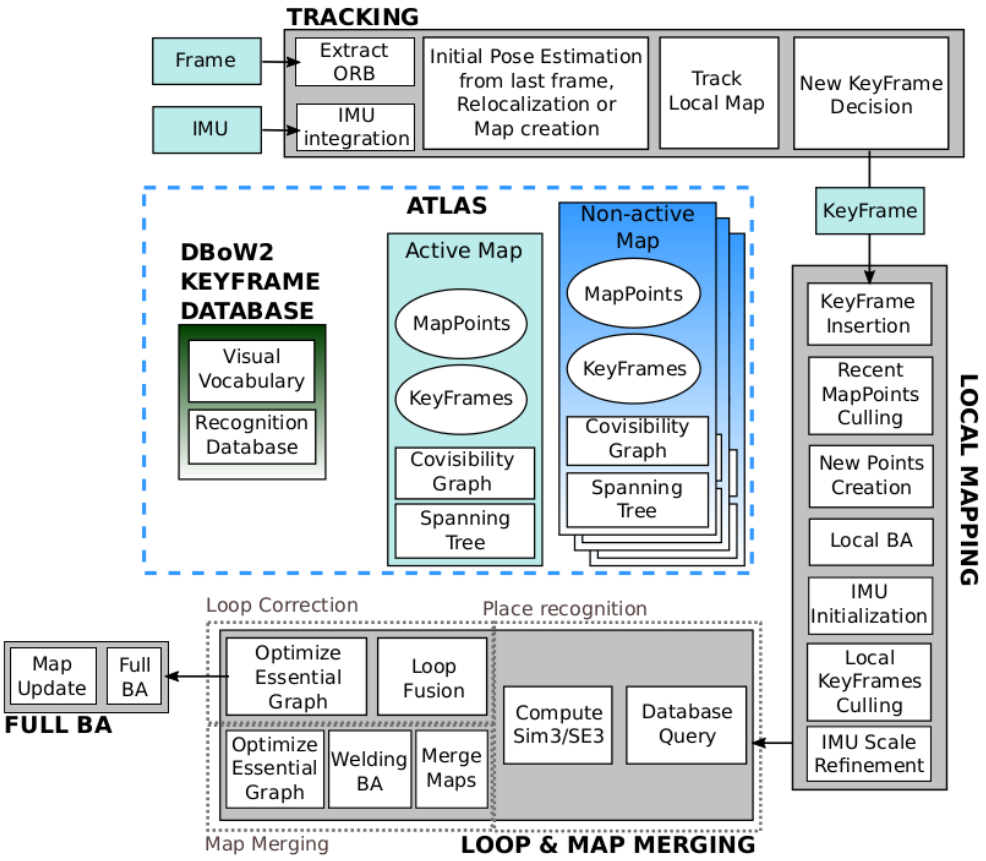


Figure 3.20: Main system components of ORB-SLAM3. The architecture comprises: an *Atlas* (managing active and non-active maps); a *Tracking thread* (sensor processing and real-time pose estimation); a *Local Mapping thread* (map refinement and IMU initialization); and a *Loop and Map Merging thread* (global consistency checks). An optional *Full Bundle Adjustment thread* performs extensive refinement. Source: Reproduced from [4].

Relocalization, Loop Closing and Map Merging.

2. **Tracking thread**: Processes sensor information and computes the pose of the current frame with respect to the Active Map in real-time, minimizing the reprojection error of the matched map features. It also decides wheter the current frame becomes a new keyframe. In Visual-Inertial mode, the body velocity and IMU biases are estimated by including the inertial residuals in the optimization. When tracking is lost, the Tracking thread tries to relocalizes the current frame in all Atlas' maps. If relocalized, tracking is resumed, switching the Active Map if needed. Otherwise, after a certain time, the Active Map is stored as non-active, and a new Active Map is initialized from scratch.

3. **Local Mapping thread**: Adds keyframes and points to the Active Map, removes the redundant ones, and refines the map using Visual or Visual-Inertial Bundle Adjustment, operating in a local window of keyframes close to the current frame.

4. **Loop and Map Merging thread**: Detects common regions between the Active Map and the whole Atlas at keyframe rate. If the common area belongs to the Active Map, it performs loop correction; if it belongs to a different map, both maps are seamlessly merged into a single one, that becomes Active Map. After a loop correction, a Full Bundle Adjustment is launched in an independent thread to further refine the map without affecting real-time performance

3.4.3 Fundamentals

While in pure Visual SLAM the estimated state only includes the current camera pose, in Visual-Inertial SLAM, additional variables need to be computed. These are the body pose $T_i = [R_i, p_i] \in \mathbf{SE}(3)$ and velocity v_i in the world frame, the gyroscope and accelerometer biases (b_i^g and b_i^a), which are assumed to evolve according to a Brownian motion. In this stochastic framework, Brownian motion models the temporal instability of the IMU sensors, often referred to as a Random Walk process. Mathematically, the biases are treated as time-varying states whose derivative is driven by a zero-mean Gaussian white noise. This characterization is essential for the MAP estimation, as it allows the optimization thread to continuously refine the bias estimates, accounting for environmental factors like thermal fluctuations that cause the sensor's zero-point to drift over time. This leads to the state vector:

$$\mathbf{S}_i \doteq \{T_i, v_i, b_i^g, b_i^a\} \tag{3.42}$$

The state vector $\mathbf{S}_i$ is defined in the manifold space, where $\mathbf{R}_i$ belongs to the Special Orthogonal Group $\mathbf{SO}(3)$, representing the 3D rotation. Formally, $\mathbf{R} \in \mathbf{SO}(3)$ is a $3x3$ matrix satisfying $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $det(\mathbf{R}) = 1$. The pose $T_i = [R_i, p_i]$ is an element of the Special Euclidean Group $\mathbf{SE}(3)$, which combines rotation and translation into a $4x4$ transformation matrix, such that:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ 0 & 1 \end{bmatrix}$$

In this tightly-integrated mode, $\mathbf{SE}(3)$ is used because the IMU provides the metric scale, rendering the world coordinates physical and fixed.

For Visual-Inertial SLAM, preintegrate IMU measurements between consecutive visual frames, i and $i+1$, obtain preintegrated rotation ($\Delta\mathbf{R}_{i,i+1}$), velocity ($\Delta\mathbf{v}_{i,i+1}$) and position ($\Delta\mathbf{p}_{i,i+1}$) measurements, as well a covariance matrix $\Sigma_{\mathcal{I}_{i,i+1}}$ for the whole measurement vector [71].

Given these preintegrated terms and states S_i and S_{i+1}, the definition of inertial residual ($\mathbf{r}_{\mathcal{I}_{i,i+1}}$) from [72]:

$$\mathbf{r}_{\mathcal{I}_{i,i+1}} = \left[\mathbf{r}_{\Delta\mathbf{R}_{i,i+1}}, \mathbf{r}_{\Delta\mathbf{v}_{i,i+1}}, \mathbf{r}_{\Delta\mathbf{p}_{i,i+1}}\right] \tag{3.43}$$

where

$$\mathbf{r}_{\Delta\mathbf{R}_{i,i+1}} = Log\left(\Delta\mathbf{R}_{i,i+1}^T \mathbf{R}_i^T \mathbf{R}_{i+1}\right)$$

$$\mathbf{r}_{\Delta\mathbf{v}_{i,i+1}} = \mathbf{R}_i^T\left(\mathbf{v}_{i+1} - \mathbf{v}_i - \mathbf{g}\Delta t_{i,i+1}\right) - \Delta\mathbf{v}_{i,i+1}$$

$$\mathbf{r}_{\Delta\mathbf{p}_{i,i+1}} = \mathbf{R}_i^T\left(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{i,i+1} - \frac{1}{2}\mathbf{g}\Delta t^2\right) - \Delta\mathbf{p}_{i,i+1}$$

and $Log : \mathbf{SO}(3) \to \mathbb{R}^3$ map from the Lie group to the vector space. Together with inertial residuals, it also is used reprojection errors $\mathbf{r}_{ij}$ between frame i and 3D point j at position $\mathbf{x}_j$:

$$\mathbf{r}_{ij} = \mathbf{u}_{ij} - \Pi\left(\mathbf{T}_{CB}\mathbf{T}_i^{-1} \oplus \mathbf{x}_j\right) \tag{3.44}$$

where $\Pi : \mathbb{R}^3 \to \mathbb{R}^n$ is the projection function for the corresponding camera model, $\mathbf{u}_{ij}$ is the observation of point j at image i, having a covariance matrix Σ_{ij}, $\mathbf{T}_{CB} \in \mathbf{SE}(3)$ stands for the rigid transformation from body-IMU to camera (left or right), known from calibration, and $\oplus$ is the transformation operation of $\mathbf{SE}(3)$ group over $\mathbb{R}^3$ elements.

Combining inertial and visual residual terms, Visual-Inertial ORB-SLAM3 can be posed a keyframe-based minimization problem [73]. Given a set of $k+1$ frames and its state $\bar{S}_k \doteq \{S_0 \cdots S_k\}$, and a set of l 3D points and its state $\mathcal{X} \doteq \{\mathbf{x}_0 \cdots \mathbf{x}_{l-1}\}$, the visual-inertial optimization problem can be stated as:

$$\min_{\bar{S}_k, \mathcal{X}} \left(\sum_{i=1}^{k} \left\| \mathbf{r}_{\mathcal{I}_{i-1,i}} \right\|^2_{\Sigma^{-1}_{\mathcal{I}_{i,i+1}}} + \sum_{j=0}^{l-1} \sum_{i \in \mathcal{K}^j} \rho_{\text{Hub}}\left(\left\| \mathbf{r}_{ij} \right\|_{\Sigma_{ij}^{-1}} \right) \right) \tag{3.45}$$

where $\mathcal{K}^j$ is the set of keyframes observing 3D point j. This optimization may be

outlined as the factor-graph shown in figure 3.21:

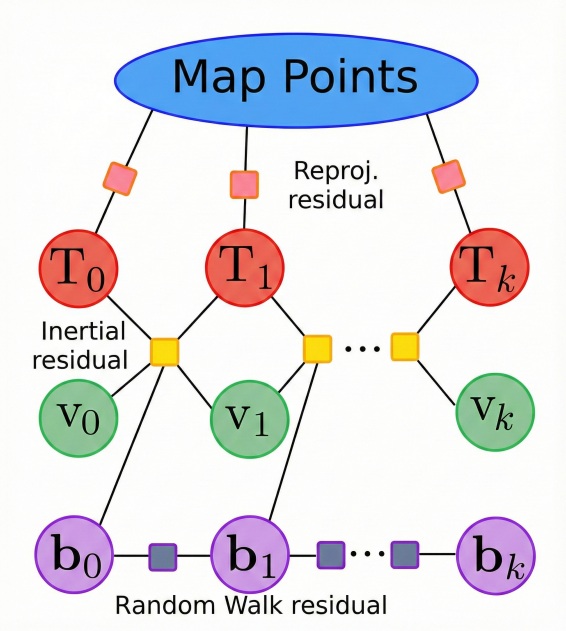


Figure 3.21: Factor graph representation of the Visual-Inertial ORB-SLAM3 fundamentals. Source: Author's own creation based on [4].

The factor graph forms a chain of inertial constraints linking consecutive states, while reprojection factors connect poses to observed 3D map points. The random walk terms maintain temporal consistency of the IMU biases. Due to perceptual aliasing, dynamic objects, or incorrect data association, a robust Huber kernel (ρ_{Hub}) is used to reduce the influence of spurious matches. The Huber loss behaves quadratically for small residuals, preserving optimality under Gaussian noise assumptions, while transitioning to linear growth for large residuals.

Notably, the Huber kernel is applied exclusively to visual reprojection residuals. Inertial residuals do not require robustification, as IMU measurements are temporally ordered and free from association ambiguities.

3.4.4 Place Recognition

To achieve long-term data association for Relocalization and Loop detection, ORB-SLAM uses bag-of-words Place Recognition system [30], [74].

DBoW2 builds a database of keyframes with their bag-of-words vectors, and given a query image is able to efficiently provide the most similar keyframes according to their bag-of-words. The elementary operation of all the geometrical verification steps consists in checking whether is an ORB keypoint inside an image window whose descriptor of a map point, using a threshold for the Hamming distance between them.

The Hamming distance enables extremely efficient matching through bitwise operations, making it suitable for real-time SLAM. In ORB-SLAM3, descriptor matching using a Hamming distance threshold constitutes the elementary operation for short-term track-

ing, mid-term map association, and long-term Place Recognition. Additionally, a ratio test with respect to the second-best match is employed to reject ambiguous correspondences, further improving robustness.

The steps of Visual-Inertial ORB-SLAM3 algorithm of Place Recognition are:

1. **DBoW2 candidate keyframes**: The Atlas DBoW2 database is queryed with the active keyframe K_a to retrieve the three most similar keyframes, excluding keyframes covisible with K_a. Each matching candidate for Place Recognition is referred as K_m.

2. **Local window**: For each K_m, a local window is defined so that includes K_m, its best covisible keyframes, and the map points observed by all of them. The DBoW2 direct index provides a set of putative matches between keypoints in K_a and in the local window keyframes.

3. **3D aligning transformation**: The transformation T_{am} is computed using RANSAC, that better aligns the map points in K_m local window with those of K_a. In monocular-inertial, when the map is still not mature, $\mathbf{T}_{am} \in Sim(3)$ is computed, otherwise $\mathbf{T}_{am} \in SE(3)$. In both cases, Horn algorithm [75] is used, so that a minimal set of three $3D-3D$ matches to find each hypotesis for $\mathbf{T}_{am}$. The putative matches that, after transforming the map point in K_a by $\mathbf{T}_{am}$, achieve a reprojection error in K_a bellow a threshold, give a positive vote to the hypotesis. The Similarity Group $\mathbf{Sim}(3)$ extends $\mathbf{SE}(3)$ by adding a scale factor $\mathbf{s} \in \mathbb{R}^+$, such that:

$$\mathbf{S} = \begin{bmatrix} \mathbf{sR} & \mathbf{p} \\ 0 & 1 \end{bmatrix}$$

4. **Guided matching refinement**: All the map points in the local window are transformed with $\mathbf{T}_{am}$ to find more matches with the keypoints in K_a. The search is also reversed, finding matches for K_a map points in all the keyframes of the local window. Using all the matchings found $\mathbf{T}_{am}$ is refined by non-linear optimization, where the goal function is the bidirectional reprojection error, using Huber influence function to provide robustness to spurious matches. If the number of inliers after the optimization is over a threshold, a second iteration of guided matching and non-linear refinement is launched using a smaller image search window.

5. **Verification in three covisible keyframes**: To avoid false positives, DBoW2 waited for Place Recognition to fire in three consecutives keyframes, delaying or missing Place Recognition. To verify Place Recognition, two keyframes covisible with K_a are searched in the active part of the map where the number of matches with points in the local window is over a threshold. If they are not found, the validation is further tried with the new incoming keyframes, without requiring the bag-of-words to fire again. The validation continues until three keyframes verify $\mathbf{T}_{am}$, or two consecutive new keyframes fail to verify it.

6. **VI Gravity direction verification**: In the visual-inertial case, if the Active Map is mature, it is estimated that $\mathbf{T}_{am} \in SE(3)$. It is further checked whether the pitch and roll angles are bellow a threshold to definitely accept the Place Recognition hypotesis.

3.4.5 Map Merging

When a successful Place Recognition produces Multi-map data association between keyframe K_a in the Active Map M_a, and a matching keyframe K_m from a different map is stored in the Atlas M_m, with an alignment transformation $\mathbf{T}_{am}$, it is launched a Map Merging operation. In this process, special care must be taken to ensure that the information in M_m can be promptly reused by the Tracking thread to avoid map duplication. For this, it is proposed to bring the M_a map into M_m reference. As M_a may contain many elements and merging them might take a long time, merging is split in two steps: First, the merge is performed in a welding window defined by the neighbours of K_a and K_m in the covisibility graph; and in a second stage, the correction is propagated to the rest of merged map by a Pose-graph optimization.

The detailed steps of Visual-Inertial ORB-SLAM3 Map Merging are:

1. **Visual-Inertial window assembly**: If the Active Map is mature, it is applied the available $\mathbf{T}_{am} \in SE(3)$ to map M_a before its inclusion in welding window. If the Active Map is not mature, M_a is aligned using the available $\mathbf{T}_{am} \in Sim(3)$.

2. **Merging Maps**: Maps M_a and M_m are fused together to become the new Active Map. To remove duplicated points, matches are actively searched for M_a points in M_m keyframes. For each match, the point from M_a is removed, and the point in M_m is kept accumulating all the observations of the removed point. The covisibility and essential graphs [27] are updated by the addition of edges connecting keyframes from M_m and M_a thanks to the new mid-term point associations found.

3. **Visual-Inertial welding Bundle Adjustment**: Poses, velocities and biases of keyframes K_a and K_m and their five last temporal keyframes are included as optimizable. These variables are related by IMU preintegration terms, as shown in figure 3.22. For M_m, the keyframe immediately before the local window is included but fixed, while for M_a the similar keyframes is included but its pose remains optimizable. All map points seen by the above mentioned keyframes are optimized, together with poses from K_m and K_a covisible keyframes. All keyframes and points are related by means of reprojection error.

4. **Essential-graph optimization**: A Pose-graph optimization is performed using the essential graph of the whole merged map, keeping fixed the keyframes in the welding area. This optimization propagates corrections from the welding window to the rest of the map.

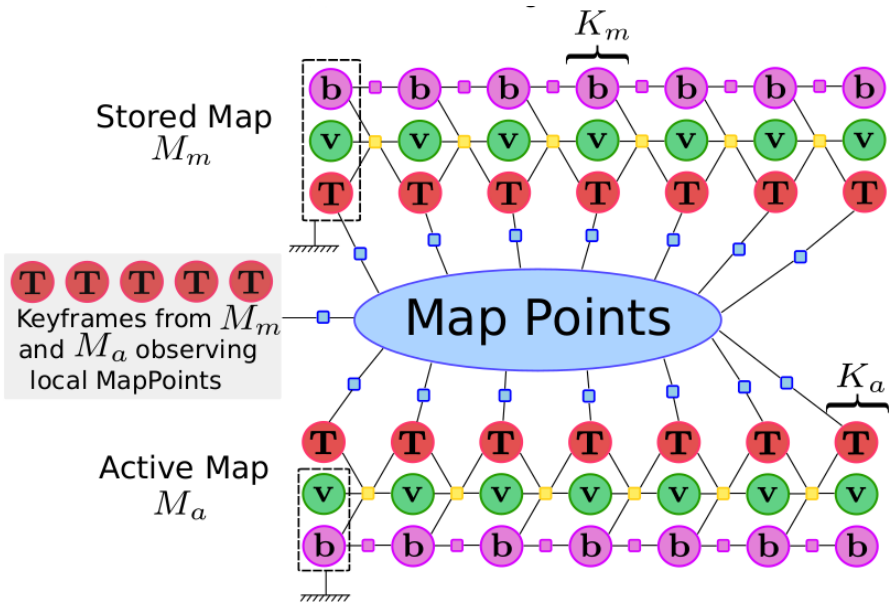


Figure 3.22: Factor graph representation of the Visual-Inertial welding Bundle Adjustment in ORB-SLAM3. Circular nodes represent state variables associated with keyframes, including body poses $T_k \in SE(3)$ (red), linear velocities v_k (green), and inertial sensor biases b_k (purple). Square nodes denote residual factors: inertial residuals from IMU preintegration (yellow), bias random walk residuals modeling Brownian motion (purple), and prior residuals used to anchor the optimization (green). The blue elliptical node labeled *Map Points* represents the set of 3D landmarks jointly observed by keyframes from the active map M_a and the stored map M_m, with reprojection residuals connecting landmarks to the corresponding pose nodes. Dashed boxes indicate fixed parameters during optimization. In particular, keyframes preceding the welding window in the stored map are fixed to provide a stable reference, while keyframes in the active map remain optimizable, allowing corrections to be consistently propagated across maps. Source: reproduced from [4].

3.4.6 Loop Closing

Loop Closing correction algorithm is analogous to Map Merging, but in a situation where both keyframes matched by Place Recognition belong to the Active Map. A welding window is assembled from the matched keyframes, and point duplicates are detected and fused creating new links in the covisibility and essential graphs. The next step is a Pose-graph optimization to propagate the loop correction to the rest of the map. The final step is a global Bundle Adjustment to find the Maximum-a-Posteriori estimate after considering the loop closure and long-term matches.

In the Visual-Inertial case, the global Bundle Adjustment is only performed if the number of keyframes is below a threshold to avoid a huge computational cost.

The experimental results show that ORB-SLAM3 is capable of effectively exploit short-term, mid-term, long-term and multi-map data associations [4]. Also, these results suggest that regarding accuracy, the capability of using all these types of data association overpowers other choices such as using direct methods instead of features, or performing keyframe marginalization for local Bundle Adjustment, instead of assuming an outer set of static keyframes.

The main failure case of ORB-SLAM3 is low-texture environments [4]. Direct methods are more robust to low-texture, but are limited to short-term [76] and mid-term [77] data association. On the other hand, matching feature descriptors successfully solves long-term and multi-map data association, but seems to be less robust for tracking than Lucas-Kanade, that uses photometric techniques adequate for that four data association problems.

Chapter 4

The POSEIDON NAV Project

This chapter introduces The **POSEIDON NAV™** [1] Project, a biologically inspired, ROS-native framework conceived to address the limitations of 2D and unimodal SLAM pipelines in perceptually degraded underwater environments. Grounded in neuro-inspired computation, POSEIDON NAV operationalizes a cognitive mapping metaphor by integrating a *NeoSLAM-like* neocortical front-end for multimodal perceptual encoding with a *DolphinSLAM-like* hippocampal back-end for 3D experience-based mapping. In doing so, the project provides a unified architecture that prioritizes perceptual robustness and

[1] The name and logo **POSEIDON NAV™** were created by the author to identify the project described in this dissertation.

global map consistency under sensing distortions that routinely challenge purely geometric solutions.

The system is engineered around two design commitments. First, **multimodal perception**: visual and acoustic sensing streams are jointly encoded into a sparse, bitwise-fused representation whose matching is based on SDR overlap. Second, **full 3D mapping**: pose inference and map construction are performed in three spatial dimensions using a 3D experience graph whose relaxation extends RatSLAM's Experience Map optimization. Beyond algorithmic design, POSEIDON NAV provides a reproducible ROS implementation with deterministic execution, explicit parameterization, and traceable data flow between perception, event generation, and mapping.

To make this architecture operational and experimentally auditable, the remainder of this chapter details PoseidonSLAM as the core SLAM method implemented within the POSEIDON NAV ROS stack, emphasizing the neocortical front-end, the event-generation interface, and the hippocampal back-end responsible for 3D experience-graph mapping.

4.1 Relation to RatSLAM, DolphinSLAM, and NeoSLAM

PoseidonSLAM inherits the hippocampal back-end principles of RatSLAM: a continuous attractor Pose Cell Network stabilized by excitation and inhibition (cf. RatSLAM-like attractor stabilization dynamics, Eq. (3.1)) and an Experience Map updated online and corrected by relaxation after loop closure (cf. Eq. (3.7)). As in DolphinSLAM, this paradigm is extended to 3D dead-reckoning and experience embedding (cf. DolphinSLAM 3D path integration, Eq. (3.10)–(3.11)).

PoseidonSLAM adopts NeoSLAM's core neocortical idea: replace intensity templates / SAD comparisons (RatSLAM LVC) with SDR-based sparse matching and temporal context (cf. NeoSLAM overlap and match, Eq. (3.29)–(3.30)). PoseidonSLAM extends this from visual-only to multimodal (vision + acoustics) while preserving the computationally cheap union operation (cf. NeoSLAM union, Eq. (3.38)).

Unlike DolphinSLAM's FAB-MAP probabilistic BoW matching (where LVC activity reflects match probability), PoseidonSLAM's place identity is produced by sequence-aware sparse inference over SDRs, naturally tolerant to partial corruption and modality dropouts, and emitted as a stable discrete identity (not a probability).

This lineage motivates a two-part architecture in which neocortical place evidence is generated continuously, while hippocampal map updates are driven discretely by experience events; Fig. 4.1 summarizes this data flow end-to-end.

4.2 PoseidonSLAM

PoseidonSLAM is a neuro-inspired SLAM framework for AUV localization in perceptually degraded underwater conditions. It encodes visual and acoustic sensory streams into sparse, *temporally contextualized* place representations and uses them to build a 3D topological experience graph supported by dead-reckoning and loop closure correction. In contrast to geometric pipelines focused on dense metric reconstruction, PoseidonSLAM emphasizes *perceptual robustness* and *map stability* by combining Sparse Distributed Representations with sequence-aware inference and experience-graph optimization.

PoseidonSLAM targets robustness to: (i) turbidity and low-contrast imagery; (ii) sonar speckle and acoustic aliasing; (iii) modality dropouts (missing camera/sonar frames); and (iv) structural repetition (corridors/tunnels). This is achieved by using (a) overlap-based similarity on SDRs, (b) Hierarchical Temporal Memory for sequence-aware context, and (c) temporal union + gating before emitting place identities.

PoseidonSLAM is deployed within the POSEIDON NAV ROS stack, which standardizes sensor interfaces, parameter loading, and an auditable event–evidence association used for reproducible experimentation.

4.2.1 System Overview

PoseidonSLAM is organized as a neuro-inspired SLAM architecture with two tightly coupled subsystems, as illustrated in Fig. 4.1.

The **Neocortical Front-End** performs multimodal perceptual encoding and publishes a continuous stream of **LVC IDs** on `/local_view_cells`. In the current runtime, camera and sonar observations are mapped to Sparse Distributed Representations through (i) lightweight CNN feature embeddings, (ii) LSBH hashing with a fixed projection matrix, and (iii) binarization by selecting the top and bottom fraction of projection responses. Modality-specific SDRs are then fused by a bitwise union (OR) to form the sensory drive of HTM Spatial Pooler and Temporal Memory. In the default configuration, LVC identity is derived from TM winner cells with short-window temporal pooling and Top-k capping. The resulting sequence-contextualized codes are matched (or instantiated) by the LVC ID module, yielding a discrete LVC identity.

The **Hippocampal Back-End** integrates the LVC identity stream with proprioceptive dead-reckoning to construct a **3D topological experience graph** (published as `/map`). Dead-reckoning is provided by a robot-state module that fuses DVL, IMU, and depth into a 6-DoF odometry estimate (`/Odometry`), which drives 3D path integration and the 4D Pose Cell attractor dynamics (x, y, z, ψ). Map updates occur only when the **VPR Guard** consistency filter authorizes the emission of `/experience_event`, thereby regulating event density and improving global map stability under perceptual aliasing. For experimental auditability, each accepted experience is associated with a synchronized sensory snapshot (`/experience_image`), enabling deterministic event–evidence linkage at analysis time.

The remainder of this section follows the runtime causal order: the neocortical encoding

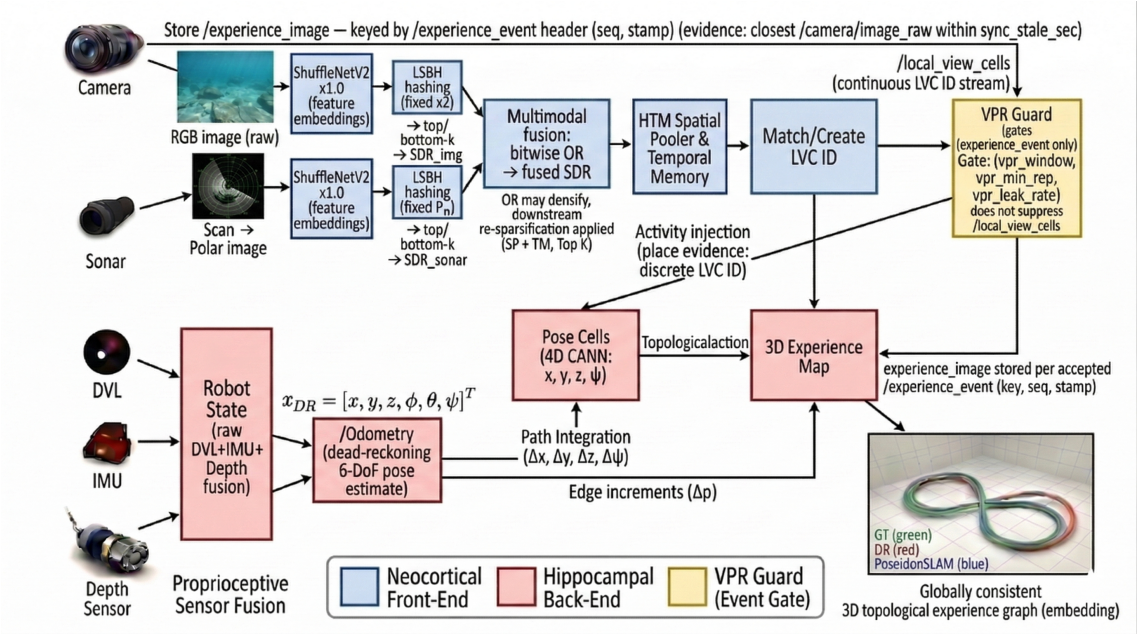


Figure 4.1: Data-flow overview of PoseidonSLAM. The Neocortical Front-End encodes camera and sonar observations using ShuffleNetV2 embeddings, applies LSBH hashing (fixed $\mathbf{P}_m$) with top/bottom-k selection to produce modality SDRs, and fuses modalities by bitwise union (OR) into a sensory drive for HTM SP/TM. LVC ID match/create a discrete LVC ID and publish a continuous identity stream on `/local_view_cells`, while VPR Guard gates only `/experience_event` (it does not suppress `/local_view_cells`). The Hippocampal Back-End integrates fused dead-reckoning from `/Odometry` with LVC evidence in a 4D Pose Cell network (x, y, z, ψ) and updates a 3D Experience Map (`/map`). Each accepted event is stored with its evidence frame `/experience_image`, keyed by the `/experience_event` header (seq, stamp) using the closest `/camera/image_raw` within `sync_stale_sec`. Source: The Author.

and continuous LVC stream (Sec. 4.2.2), the event-generation interface via VPR Guard (Sec. 4.2.3), and the hippocampal back-end mechanisms that convert gated events into a 3D experience graph (Sec. 4.2.4).

4.2.2 Neocortical Front-End

The Front-End encodes visual and acoustic observations and emits discrete LVC identifiers suitable for experience-based mapping. Each modality is mapped to a compact embedding by a lightweight CNN backbone (e.g., ShuffleNetV2), chosen for its favorable accuracy–latency trade-off under real-time and embedded constraints [12]. The resulting embeddings are projected and binarized into sparse binary codes compatible with SDR-style processing (as in NeoSLAM's pipeline [9]).

In PoseidonSLAM, an LVC denotes a discrete place identity derived from *multimodal* and *temporally consolidated* sparse codes designed to remain stable under modality-specific degradations and sensor dropouts. This differs conceptually from (i) RatSLAM's template-SAD LVCs (which inject via Eq. (3.3)), (ii) DolphinSLAM's probabilistic FAB-MAP

LVC activations, and aligns structurally with NeoSLAM's union-based stabilization (cf. Eq. (3.38)), while extending it to cross-modality fusion.

Figure 4.2 summarizes the PoseidonSLAM front-end from raw multimodal observations to the published place-identity stream. Per modality, camera and sonar inputs are embedded using a lightweight CNN backbone, projected via LSBH with a fixed matrix, and binarized by selecting top and bottom responses to yield modality SDRs. A bitwise union (OR) forms the fused sensory drive $\mathbf{s}_t^{\text{fused}}$ for the HTM Spatial Pooler and Temporal Memory. Importantly, place identity matching is performed on a TM-derived descriptor $\mathcal{D}_t$ (obtained from TM winner cells followed by temporal pooling, interval union, and Top-k capping), rather than directly on $\mathbf{s}_t^{\text{fused}}$. The LVC ID module applies the overlap-based match rule (with an optional F1 gate) to match or instantiate an LVC ID, published continuously on `/local_view_cells`. Downstream event gating (VPR Guard) acts only in the back-end by regulating `/experience_event` and does not suppress `/local_view_cells`.

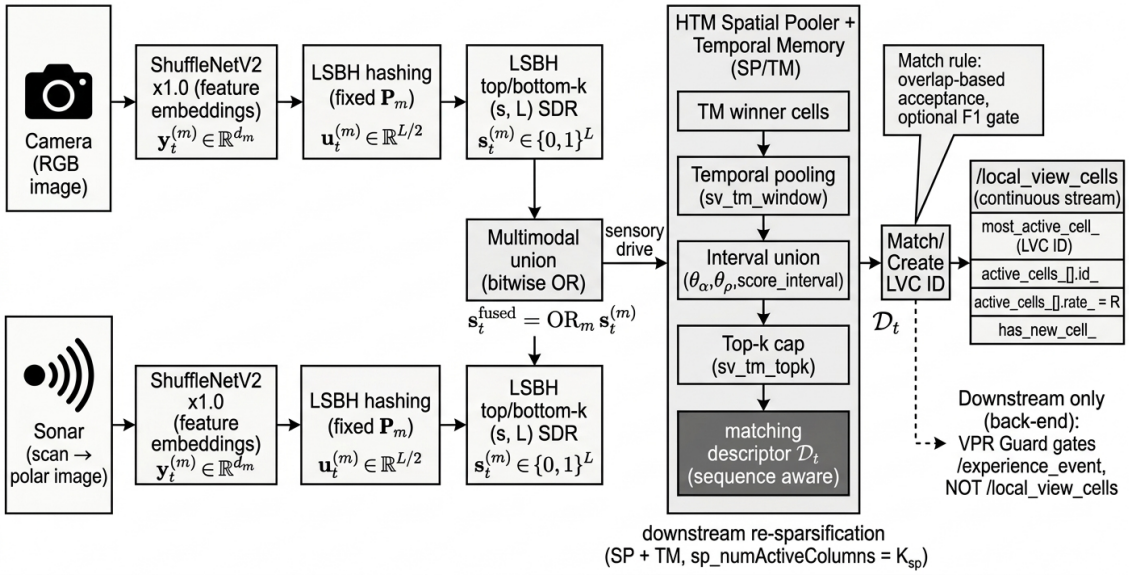


Figure 4.2: PoseidonSLAM Front-End pipeline (end-to-end): Camera and Sonar observations are embedded by a lightweight CNN (ShuffleNetV2) to produce modality embeddings $\mathbf{y}_t^{(m)} \in \mathbb{R}^{d_m}$. Each embedding is mapped by LSBH with a fixed projection $\mathbf{P}_m$, yielding responses $\mathbf{u}_t^{(m)} \in \mathbb{R}^{L/2}$ and a binary SDR $\mathbf{s}_t^{(m)} \in \{0,1\}^L$ via top/bottom-k selection. Modality SDRs are fused by a bitwise union (OR) to form the sensory drive $\mathbf{s}_t^{\text{fused}}$, which feeds HTM SP/TM. TM winner cells are consolidated into a sequence-aware matching descriptor $\mathcal{D}_t$ through temporal pooling (`sv_tm_window`), interval union ($\theta_\alpha, \theta_\rho$, `score_interval`), and Top-$k$ capping (`sv_tm_topk`), with downstream re-sparsification controlled by `sp_numActiveColumns`. LVC ID then matches/creates an LVC ID using overlap-based acceptance with optional F1 gate, publishing the continuous place-identity stream on `/local_view_cells`. VPR Guard operates only downstream in the Back-End to gate `/experience_event`, not `/local_view_cells`. Source: The Author.

Let $\mathcal{M}_t$ be the set of modalities available at time t (e.g., vision, sonar), and let $\mathbf{o}_t^{(m)}$ be a raw observation from modality $m \in \mathcal{M}_t$. Each modality is encoded into a real-valued

embedding by a backbone $f_m(\cdot)$. The resulting vector $\mathbf{y}_t^{(m)}$,

$$\mathbf{y}_t^{(m)} = f_m\Big(\mathbf{o}_t^{(m)}\Big) \in \mathbb{R}^{d_m}\,, \tag{4.1}$$

is a compact continuous descriptor that summarizes the salient perceptual content of $\mathbf{o}_t^{(m)}$ (e.g., visual appearance cues or acoustic backscatter echo-intensity signatures) in a feature space suitable for subsequent random projection and Top-k binarization.

The notation $\mathbf{y}_t^{(m)} \in \mathbb{R}^{d_m}$ means that the embedding is a d_m-dimensional real-valued vector, where d_m is the output dimensionality of the backbone for modality m (i.e., the number of scalar features produced per observation). In the current implementation, $f_m(\cdot)$ is instantiated as a lightweight CNN (ShuffleNetV2) and the embedding dimension d_m corresponds to the flattened feature tensor extracted from an intermediate convolutional block.

To obtain a Sparse Distributed Representation, PoseidonSLAM uses LSBH hashing with a fixed projection matrix. Let L denote the SDR length used by the front-end (shared with the SP), and let d_m be the embedding dimensionality for modality m. For each modality, a fixed LSBH projection $\mathbf{P}_m \in \mathbb{R}^{d_m \times (L/2)}$ (loaded from `lsbh_matrix_path`) maps the embedding $\mathbf{y}_t^{(m)}$ into a vector of linear responses $\mathbf{v}_t^{(m)} = \mathbf{P}_m^\top \mathbf{y}_t^{(m)} \in \mathbb{R}^{L/2}$, where each component reflects the signed projection of the embedding onto a random hyperplane. Let $s \in (0,1)$ be the LSBH selection ratio (`lsbh_s`), and define $k = \lfloor s(L/2) \rfloor$ as the number of selected indices per half. LSBH encodes both the strongest positive and strongest negative responses by concatenating the top-k and bottom-k indicator sets into a single binary vector of length L.

Let $\mathbb{B} \triangleq \{0,1\}$. The modality SDR is $\mathbf{s}_t^{(m)} \in \mathbb{B}^L$, with $p = L/2$, such that

$$s_{t,i}^{(m)} = \begin{cases} \mathbb{I}\Big\{\, i \in \mathrm{TopK}\Big(\mathbf{v}_t^{(m)}, k\Big)\Big\}\,, & 1 \le i \le p, \\ \mathbb{I}\Big\{\, (i-p) \in \mathrm{BottomK}\Big(\mathbf{v}_t^{(m)}, k\Big)\Big\}\,, & p < i \le 2p, \end{cases} \tag{4.2}$$

where TopK returns the indices of the k largest components of $\mathbf{v}_t^{(m)}$, BottomK returns the indices of the k smallest components, and $\mathbb{I}\{\cdot\}$ is the indicator function. The concatenation yields exactly $2k$ active bits per modality, providing a fixed-sparsity SDR used downstream in the PoseidonSLAM front-end. The resulting modality-specific SDRs are illustrated in Fig. 4.2.

The notation $s_{t,i}^{(m)}$ denotes the i-th entry of the SDR vector, for $i \in \{1, \ldots, L\}$. The indicator function $\mathbb{I}\{\cdot\}$ evaluates to 1 if its condition is true and to 0 otherwise. Therefore, Eq. (4.2) sets $s_{t,i}^{(m)} = 1$ exactly for the k indices corresponding to the largest values in $\mathbf{u}_t^{(m)}$, and $s_{t,i}^{(m)} = 0$ elsewhere. Thus, each modality code has fixed sparsity $\|\mathbf{s}_t^{(m)}\|_1 = k$ consistent with SDR processing.

This operating regime is consistent with the HTM notion of Sparse Distributed Representations. In BaMI [78], Hawkins *et al.* provide a canonical rule-of-thumb for SDR dimensionality and sparsity: "An SDR may have 2,000 bits with 2% being 1 and the

rest 0". The PoseidonSLAM default (L=2048, k=20 active SP columns) operates in a slightly sparser regime, which reflects the broader cortical sparse-representation viewpoint emphasized in the Thousand Brains Theory [79]—namely, that distributed sparse activity patterns constitute a fundamental substrate for robust representation across cortical modules. In high-dimensional SDRs, sparsity makes random overlaps unlikely, so overlap becomes a meaningful similarity signal rather than a chance coincidence; moreover, distributed redundancy across active bits yields robustness to partial corruption and missing evidence (e.g., modality dropouts) [78, 80]. In practice, overlap-based matching can be implemented as set intersection over active indices, scaling with the number of active bits ($\mathcal{O}(k)$) rather than with L for the matching step itself [78].

In the current runtime, the multimodal union is the *sensory drive* provided to the HTM framework (Spatial Pooler and Temporal Memory). Concretely, each available modality $m \in \mathcal{M}_t$ produces a binary SDR $\mathbf{s}_t^{(m)} \in \mathbb{B}^L$, and their bitwise union forms the fused HTM input

$$\mathbf{s}_t^{\text{fused}} \triangleq \bigvee_{m \in \mathcal{M}_t} \mathbf{s}_t^{(m)}, \tag{4.3}$$

where, $\triangleq$ indicates that $\mathbf{s}_t^{\text{fused}}$ "is defined as" the bitwise union of the modality SDRs. As shown in Fig. 4.2, the multimodal union provides the HTM sensory drive rather than the final matching descriptor. This fused code aggregates instantaneous evidence across sensors and is therefore robust to single-modality degradation and modality dropouts (missing frames). Note that union (OR) typically increases descriptor density; therefore, PoseidonSLAM employs conservative gating and re-esparsification downstream to mitigate union-induced densification effects (e.g., false positives).[2]

In NeoSLAM, the union operator is used to stabilize place evidence by aggregating a short sequence of SDRs over time, i.e., $\mathbf{d}_{\text{out}} = \bigvee_{i=0}^{K-1} \mathbf{d}_{t-i}$(Eq. (3.38)). In PoseidonSLAM, the same operator is first applied across modalities at time t (Eq. (4.3)) so that informative bits from any available sensor contribute to the place representation; consequently, if one modality is degraded or absent, the remaining modalities can still sustain recognition.

Importantly, the descriptor effectively used for *place identity matching* is *not* the raw fused code $\mathbf{s}_t^{\text{fused}}$. This distinction is highlighted in Fig. 4.2, where $\mathcal{D}_t$ is formed from TM winner cells and subsequent consolidation prior to LVC ID matching. By default, the Local View Cell identity (LVC ID) is derived from the Temporal Memory state. After $\mathbf{s}_t^{\text{fused}}$ is processed by the Spatial Pooler and Temporal Memory, we extract the TM *winner cells* and apply three lightweight consolidation steps: (i) temporal pooling over a short window (`sv_tm_window`), which emphasizes consistently recurring winners; (ii) interval union

[2]PoseidonSLAM enforces map stability through a three-stage mechanism: (i) a sequence-aware descriptor $\mathcal{D}_t$ derived from HTM Temporal Memory winner cells, temporally pooled over a short window and re-sparsified by Top-k capping; (ii) an optional symmetric match gate based on the Sørensen–Dice (F1) score, used alongside the primary overlap criterion to curb descriptor densification effects; and (iii) the VPR Guard consistency filter in the back-end, which authorizes `/experience_event` only after sufficient repetition of the current LVC ID within a sliding window. Thus, `/local_view_cells` remain a continuous perceptual stream, while updates to the 3D Experience Map occur only via gated `/experience_event` emissions.

(Eq. (4.5)), which accumulates evidence over short runs while similarity remains above a threshold; and (iii) a final Top-k re-sparsification (`sv_tm_topk`), which bounds descriptor density after pooling/union. We denote the resulting TM-derived binary descriptor by $\mathbf{s}_t^{\mathrm{tm}}$.

Since $\mathbf{s}_t^{\mathrm{tm}}$ depends on the current input and the recent SP/TM context, it yields a *sequence-aware* place identity while keeping the match operation overlap-based. Figure 4.3 provides a conceptual view of how TM winner cell selection is conditioned on temporal context (distal segments) and feedback (apical input), explaining the sequence-aware behavior summarized in Fig. 4.2.

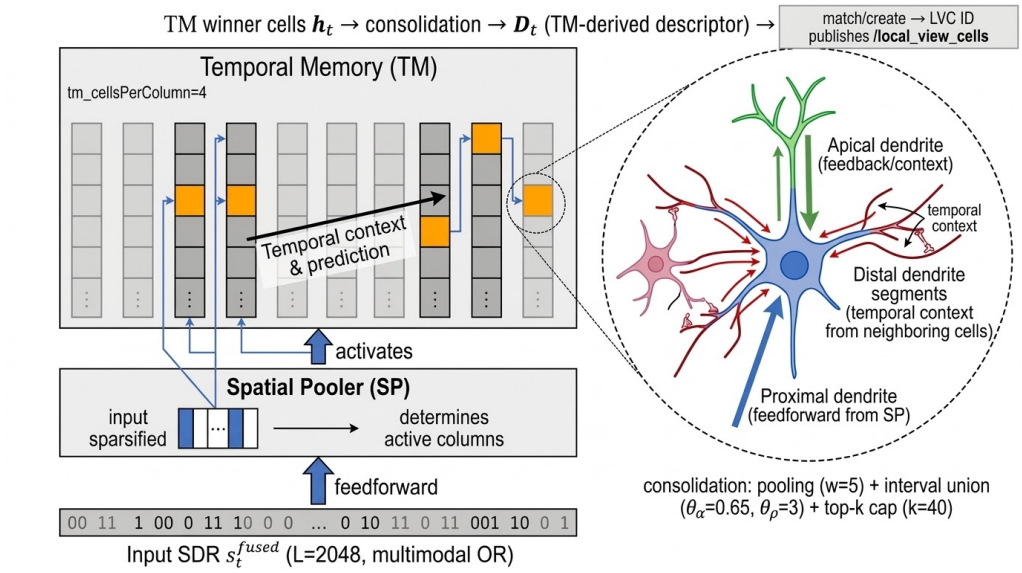


Figure 4.3: HTM-based place-identity formation in the PoseidonSLAM front-end. The fused multimodal SDR $\mathbf{s}_t^{\mathrm{fused}}$ drives the Spatial Pooler and Temporal Memory. TM winner cells (h_t) are consolidated (pooling, interval union, and Top-k cap) to form the TM-derived descriptor $\mathcal{D}_t$, which is matched/instantiated by LVC ID to produce the LVC ID published on `/local_view_cells`. Sequence-awareness arises because TM selects winner cells conditioned on temporal context (distal segments) and, conceptually, higher-level contextual feedback, not only on feedforward activation (proximal input). Source: The Author.

We denote the active-bit set of the matching descriptor by:

$$\mathcal{A}_t \doteq \{\, i \in \{1, \ldots, L\} \;:\; \mathbf{s}_t^{\mathrm{tm}}[i] = 1 \,\}. \tag{4.4}$$

To reduce sensitivity to single-frame noise and mitigate repetitive-structure segments, PoseidonSLAM optionally consolidates evidence over short runs by unioning the active-bit sets within an interval:

$$\mathcal{D}_t = \bigcup_{i=0}^{K_t-1} \mathcal{A}_{t-i}, \qquad K_t \le K_{\max}. \tag{4.5}$$

where K_t is the current run length (bounded by $K_{\max}$) determined by an interval condition (e.g., while similarity to the previous descriptor remains above a threshold). In the remainder, $\mathcal{D}_t$ denotes the descriptor used for LVC matching.

Local View Cells

Let $\mathcal{D}_t$ be the current active-bit set used for matching and let $\mathcal{D}_\ell$ be the stored prototype for a candidate LVC $\ell \in \mathcal{L}$, where $\mathcal{L}$ denotes the set of all LVC IDs currently stored by the front-end; each ℓ is associated with a stored prototype descriptor $\mathcal{D}_\ell$ used for overlap-based matching. We first define the intersection cardinality

$$I(\mathcal{D}_t, \mathcal{D}_\ell) \doteq |\mathcal{D}_t \cap \mathcal{D}_\ell| \,. \tag{4.6}$$

In PoseidonSLAM, LVC matching uses an overlap–recall similarity score

$$R(\mathcal{D}_t, \mathcal{D}_\ell) \doteq \frac{|\mathcal{D}_t \cap \mathcal{D}_\ell|}{|\mathcal{D}_t|}, \qquad 0 \le R \le 1 \,, \tag{4.7}$$

which is robust to descriptor densification because it normalizes by the current descriptor size $|\mathcal{D}_t|$. [3]

For each stored identity ℓ, PoseidonSLAM evaluates the overlap-based similarity $R(\mathcal{D}_t, \mathcal{D}_\ell)$ and selects the most consistent prototype under the primary score:

$$\ell^\star = \arg\max_{\ell \in \mathcal{L}} R(\mathcal{D}_t, \mathcal{D}_\ell) \tag{4.8}$$

A recognition is accepted only if the winning candidate satisfies a two-stage acceptance test:

$$I(\mathcal{D}_t, \mathcal{D}_{\ell^\star}) \ge I_{\min} \;\wedge\; R(\mathcal{D}_t, \mathcal{D}_{\ell^\star}) \ge \tau_R \tag{4.9}$$

where $I(\mathcal{D}_t, \mathcal{D}_{\ell^\star})$ is the absolute intersection cardinality. The first constraint $I_{\min}$ acts as an evidence floor that rejects low-support, chance overlaps in high-dimensional codes, while the second constraint τ_R enforces a minimum normalized agreement with the current descriptor. In the ROS implementation, the primary score $R(\mathcal{D}_t, \mathcal{D}_{\ell^\star})$ is reported as the LVC activation "rate" in the `/local_view_cells` stream.

If Eq. (4.9) fails, the observation is treated as novel and a new LVC ID is instantiated by storing $\mathcal{D}_t$ as a new prototype:

$$\mathcal{L} \leftarrow \mathcal{L} \cup \{\ell_{\text{new}}\}, \qquad \mathcal{D}_{\ell_{\text{new}}} \leftarrow \mathcal{D}_t \tag{4.10}$$

To mitigate repetitive-structure segments, PoseidonSLAM adopts an *event-level* emission gate that prevents overly frequent or weakly supported recognitions from driving the

[3] **Equivalence with Hamming Distance under Top-k Constraints:** By representing descriptors as sparse binary vectors $\mathbf{x}, \mathbf{y} \in \{0,1\}^L$ with associated active-bit sets $\mathcal{A}$ and $\mathcal{B}$, the Hamming distance is formally defined as the cardinality of their symmetric difference: $d_H(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} \oplus \mathbf{y}\|_1 = |\mathcal{A} \Delta \mathcal{B}| = |\mathcal{A}| + |\mathcal{B}| - 2|\mathcal{A} \cap \mathcal{B}|$. In PoseidonSLAM, descriptors are subject to Top-k capping (e.g., via `sv_tm_topk`), ensuring that $|\mathcal{A}| \approx |\mathcal{B}| \approx k$. Under this fixed-sparsity regime, the Hamming distance simplifies to a linear function of the intersection: $d_H = 2(k - |\mathcal{A} \cap \mathcal{B}|)$. Consequently, maximizing the asymmetric recall R (or overlap) is strictly equivalent to minimizing the Hamming distance. The match criterion $R \ge \tau_R$ can thus be analytically mapped to a Hamming threshold $d_H \le 2k(1 - \tau_R)$, subject to the absolute evidence floor $|\mathcal{A} \cap \mathcal{B}| \ge I_{\min}$.

map update.

Importantly, in the current ROS implementation this gate is *not* applied by suppressing LVC publication: the neocortical front-end always publishes `/local_view_cells`. Instead, the effective gating is enforced *downstream* in the hippocampal back-end, where the **VPR Guard** module decides whether an incoming LVC ID stream is sufficiently consistent to authorize the generation of an experience-driving event (i.e., allowing `/experience_event` to be published).

Worked example: Local View Cell creation from camera and sonar inputs

Goal: Show, step by step, how PoseidonSLAM converts camera + sonar observations into a discrete *LVC ID* . We keep the **default parameters** of the real system (L=2048, s=0.25, `sp_numActiveColumns`=20, `sv_tm_window`=7, `sv_tm_topk`=15, $I_{\min}$=12, τ_R=0.30, τ_{F_1}=0.55), but use a **illustrative SDR** (L=16) only to make the *LSBH binarization* and the *multimodal OR* visually explicit. The subsequent SP/TM and LVC ID steps are described with the real (default) dimensions.

(1) Embeddings (continuous). Each sensor frame is mapped to a compact real vector by the CNN:

$$\mathbf{y}_t^{(m)} = f_m(\mathbf{o}_t^{(m)}) \in \mathbb{R}^{d_m}, \qquad m \in \{\text{cam}, \text{son}\} \quad \text{(Eq. (4.1))}.$$

Interpretation: $\mathbf{y}_t^{(m)}$ is a learned *summary* of the camera or sonar observation.

(2) LSBH hashing: real-valued responses then binarization. LSBH projects each embedding with a fixed matrix $\mathbf{P}_m$, producing a real-valued response vector

$$\mathbf{v}_t^{(m)} = \mathbf{P}_m^\top \mathbf{y}_t^{(m)} \in \mathbb{R}^p \quad \text{(Eq. (4.2))}.$$

Then, only the k largest and k smallest responses are kept as a sparse binary code.

Illustrative example: Let L=16, p=8, s=0.25 $\Rightarrow$ k=2.

Camera responses:

$$\mathbf{v}_t^{(\text{cam})} = [\,0.9,\ -1.2,\ 0.1,\ -0.8,\ 2.1,\ -0.4,\ 0.7,\ -1.5\,].$$

Top-k indices are $\{5, 1\}$ (values 2.1 and 0.9), bottom-k indices are $\{8, 2\}$ (values -1.5 and -1.2). Thus,

$$\mathcal{A}_{\text{top}}^{(\text{cam})} = \{5, 1\}, \qquad \mathcal{A}_{\text{bottom}}^{(\text{cam})} = \{8, 2\},$$

$$\mathcal{A}^{(\text{cam})} = \mathcal{A}_{\text{top}}^{(\text{cam})} \cup \{p + j : j \in \mathcal{A}_{\text{bottom}}^{(\text{cam})}\} = \{1, 5, 10, 16\}.$$

Binary view:

$$\mathbf{s}_t^{(\text{cam})} = [1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1].$$

Sonar responses:

$$\mathbf{v}_t^{(\text{son})} = [\,0.2,\ -0.9,\ 1.5,\ -1.1,\ 0.4,\ -0.2,\ 0.8,\ -1.7\,].$$

Top-k indices are $\{3, 7\}$ (values 1.5 and 0.8), bottom-k indices are $\{8, 4\}$ (values -1.7 and -1.1). Thus,

$$\mathcal{A}^{(\text{son})} = \{3, 7, p+8, p+4\} = \{3, 7, 12, 16\},$$

$$\mathbf{s}_t^{(\text{son})} = [0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1].$$

Interpretation: the CNN produces continuous embeddings; LSBH turns them into real-valued responses; only extreme responses survive as a sparse binary SDR.

(3) Multimodal union (sensory drive). Camera and sonar SDRs are fused by a bitwise OR:

$$\mathbf{s}_t^{\text{fused}} = \bigvee_{m \in \mathcal{M}_t} \mathbf{s}_t^{(m)} \quad \text{(Eq. (4.3))}.$$

In the illustrative example,

$$\mathbf{s}_t^{\text{fused}} = \mathbf{s}_t^{(\text{cam})} \vee \mathbf{s}_t^{(\text{son})} = [1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1].$$

Interpretation: any active bit from camera *or* sonar survives, so the system can tolerate partial degradations in a single modality.

(4) SP/TM and temporal descriptor (sequence awareness). The fused SDR $\mathbf{s}_t^{\text{fused}}$ is still a raw sensory code. The Spatial Pooler enforces a fixed sparsity, selecting exactly `sp_numActiveColumns` $= 20$ active columns:

$$\mathcal{C}_t = \text{SP}\left(\mathbf{s}_t^{\text{fused}}\right), \qquad |\mathcal{C}_t| = 20.$$

These columns feed the Temporal Memory (TM), which produces *winner cells* $\mathcal{W}_t$ conditioned not only on the current input but also on recent context. To make the descriptor stable, PoseidonSLAM aggregates evidence across a short window (`sv_tm_window`=7) and then caps the descriptor to `sv_tm_topk` $= 15$ active indices:

$$\mathcal{D}_t = \text{cap}_{15}\left(\text{IU}\left(\bigcup_{i=0}^{6} \mathcal{A}_{t-i}\right)\right) \quad \text{(Eq. (4.5))}.$$

Didactic intuition: if the union over the last 7 steps yields, say, 18 active indices, the Top-k cap keeps only the 15 most frequent/stable ones. Thus $\mathcal{D}_t$ is a compact, sequence-aware descriptor used for matching.

(5) LVC ID matching and assignment. The current descriptor $\mathcal{D}_t$ is compared against stored prototypes $\{\mathcal{D}_\ell\}$. Two quantities are computed (Eqs. (4.6)–(4.7)):

$$I = |\mathcal{D}_t \cap \mathcal{D}_\ell| \quad \text{(absolute evidence)}, \qquad R = \frac{|\mathcal{D}_t \cap \mathcal{D}_\ell|}{|\mathcal{D}_t|} \quad \text{(normalized overlap)}.$$

The recognition is accepted only if both $I \geq I_{\min}$ and $R \geq \tau_R$ (Eq. (4.9)). *Example:* if $|\mathcal{D}_t| = 15$ and the best prototype shares $I = 12$ indices, then $R = 12/15 = 0.80$ and the match is accepted. If any criterion fails, a new LVC is created and stored (Eq. (4.10)). When the optional F1 gate is enabled, the match is further required to satisfy $F_1 \geq \tau_{F_1}$.

(6) What gets published (and what does not). The front-end publishes a *continuous identity stream* on `/local_view_cells` containing:

$$\texttt{most\_active\_cell\_} = \ell_t, \qquad \texttt{active\_cells\_}[0].\texttt{rate\_} = R,$$

plus the standard ROS header (seq, stamp) for ordering and auditability. The descriptor $\mathcal{D}_t$ itself is *not* serialized; it remains internal to the HTM pipeline.

Final remark (link to VPR Guard). Only the emitted id sequence $\{\ell_t\}$ is used by VPR Guard in the back-end to gate `/experience_event`; the stream `/local_view_cells` is never suppressed.

4.2.3 VPR Guard: Temporal-Consistency Gating of Experience Events

Even under conservative front-end place matching (cf. Eq. (4.9)), spurious recognitions can still arise due to perceptual aliasing, transient sensing artifacts, and modality-specific degradations. PoseidonSLAM therefore employs a back-end temporal consistency filter, denoted **VPR Guard**, whose role is to decide whether the incoming *stream of LVC identities* is sufficiently stable to authorize an experience-driving update (i.e., publication of `/experience_event`). Importantly, VPR Guard operates *only* on repetition patterns of the emitted LVC IDs; it does not access descriptor-level quantities such as $I(\cdot,\cdot)$, $R(\cdot,\cdot)$, or $F_1(\cdot,\cdot)$, which are computed in the front-end to select ℓ_t.

Let $\ell_t \in \mathcal{L}$ denote the most-active LVC ID published at time step t, where $\mathcal{L}$ is the set of instantiated LVC identities. Let $w \in \mathbb{N}$ be the VPR Guard window size. We define the repetition count of the current identity within the last w emissions as

$$C_t(\ell_t) \triangleq \sum_{i=0}^{w-1} \mathbb{I}\{\ell_{t-i} = \ell_t\}, \tag{4.11}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function (equal to 1 if its predicate is true, and 0 otherwise).

Given a minimum repetition threshold $r \in \{1, \ldots, w\}$, the VPR Guard decision is the Boolean gate:

$$g_t^{\mathrm{VPR}} \triangleq \mathbb{I}\{C_t(\ell_t) \geq r\}. \tag{4.12}$$

An experience-driving event is authorized when $g_t^{\mathrm{VPR}} = 1$.

To avoid indefinite suppression over long repetitive segments or under persistently marginal identity streams, VPR Guard may enable a *leak* rule that forces periodic acceptance after a bounded number of consecutive rejections. Let q_t be the run length of consecutive rejections,

$$q_t = \begin{cases} q_{t-1} + 1, & \text{if } g_t^{\mathrm{VPR}} = 0, \\ 0, & \text{if } g_t^{\mathrm{VPR}} = 1, \end{cases} \tag{4.13}$$

and let $L_{\max} \in \mathbb{N}$ be the maximum allowed rejection run. The leaked decision is then

$$\tilde{g}_t^{\mathrm{VPR}} \triangleq g_t^{\mathrm{VPR}} \vee \mathbb{I}\{q_t \geq L_{\max}\}, \tag{4.14}$$

so that an event is emitted either when the repetition criterion is satisfied or when the rejection run reaches $L_{\max}$.

For ablation, we define two operating modes:

$$\begin{aligned} &\textbf{VPR Guard ON:} \quad \text{authorize events using } \tilde{g}_t^{\mathrm{VPR}} \\ &\textbf{VPR Guard OFF:} \quad \tilde{g}_t^{\mathrm{VPR}} \equiv 1 \quad \forall t. \end{aligned}$$

Thus, in **OFF** mode, every emitted identity may trigger `/experience_event`, subject only to the remaining back-end conditions.

This formalization makes the ON/OFF comparison well-defined: the front-end controls *which* identity is emitted, while VPR Guard controls *whether* the resulting identity stream is sufficiently consistent to produce experience-driving events and, consequently, to update the 3D experience graph.

In our default ablation protocol, we compare **VPR Guard ON** (`require_vpr_for_event=true`, $w = 26$, $r = 3$, leak rate $L_{\max} = 20$) versus **OFF** (`require_vpr_for_event=false`). Consistent with the intended system behavior, **ON** reduces event rate and improves global map stability (lower risk of topological collapse), whereas **OFF** increases event density and may degrade global consistency by injecting weaker or less stable identity streams into the experience-map update.

Worked example: VPR Guard (default settings). Assume the default guard parameters $w = 26$, $r = 3$, $L_{\max} = 20$ (Eqs. (4.11)–(4.14)). Let the current emitted identity be $\ell_t = 12$. Assume that, within the last $w = 26$ emissions, the identity 12 appears four times. For brevity, we only show the last seven emissions:

$$[\ell_{t-6}, \ldots, \ell_t] = [12,\ 12,\ 7,\ 12,\ 5,\ 9,\ 12].$$

Thus, the repetition count in the full w-window is $C_t(\ell_t) = 4$, and

$$g_t^{\mathrm{VPR}} = \mathbb{I}\{C_t(\ell_t) \geq r\} = \mathbb{I}\{4 \geq 3\} = 1 \quad \text{(Eq. (4.12))}.$$

Therefore, the back-end authorizes `/experience_event` at time t.

Now suppose the stream becomes unstable for the next steps, with no identity reaching the repetition threshold inside the rolling w-window. For instance, let the next identities be all distinct:

$$[\ell_{t+1}, \ell_{t+2}, \ell_{t+3}, \ell_{t+4}] = [7,\ 4,\ 9,\ 6].$$

Note that some of these ids may still appear twice within the last w emissions due to recent history (e.g., $\ell_{t+1} = 7$ also appears at $\ell_{t-4} = 7$, and $\ell_{t+3} = 9$ also appears at $\ell_{t-1} = 9$),

but in all cases the repetition count remains below the threshold $r = 3$. Therefore,

$$C_{t+k}(\ell_{t+k}) < r \quad \Rightarrow \quad g_{t+k}^{\mathrm{VPR}} = 0, \qquad k = 1, \ldots, 4.$$

Let q_{t+k} denote the run length of consecutive rejections (Eq. (4.13)). If $g_{t+1}^{\mathrm{VPR}} = \cdots = g_{t+20}^{\mathrm{VPR}} = 0$, then $q_{t+20} = 20 = L_{\max}$ and the leak rule forces acceptance:

$$\tilde{g}_{t+20}^{\mathrm{VPR}} = g_{t+20}^{\mathrm{VPR}} \ \vee \ \mathbb{I}\{q_{t+20} \geq L_{\max}\} = 0 \vee 1 = 1 \quad \text{(Eq. (4.14))}.$$

Therefore, an experience event is emitted at time $t{+}20$ even though $g_{t+20}^{\mathrm{VPR}} = 0$ (and the rejection counter is reset thereafter).

Interpretation. With VPR Guard ON, an experience event is emitted only when an identity repeats sufficiently (r times in a w-window), unless the leak rule prevents indefinite suppression. With VPR Guard OFF, the gate is bypassed and every emitted identity may trigger an event (subject to the remaining back-end conditions).

Having defined the event-authorization mechanism, we now detail how gated experience events, together with fused dead-reckoning, drive the 4D pose-cell dynamics and the construction/relaxation of the 3D experience graph.

4.2.4 Hippocampal Back-End

The back-end fuses dead-reckoning and LVC cues to construct a globally consistent *3D topological experience graph*. This design follows the DolphinSLAM Experience Map paradigm [49], while using neocortical LVC IDs as the primary perceptual drive.

PoseidonSLAM uses dead-reckoning increments provided by the ROS `/Odometry` stream, which is treated as a global-frame pose estimate in the current runtime. In our ROS stack, the `/Odometry` stream is the output of a dedicated robot-state fusion module that combines proprioceptive sensing, namely DVL (`/dvl_linkquest`), IMU (`/imu_adis_ros`) and depth (`/depth_sensor`), and publishes a dead-reckoning estimate in the global frame.[4]

Accordingly, 3D path integration in the current runtime is realized by discrete differencing of successive odometry poses. Let $\mathbf{p}_t^{\mathrm{odom}} \in \mathbb{R}^3$ denote the translational component of the fused `/Odometry` message at time t (expressed in the odometry/world frame adopted by the system). The incremental dead-reckoning displacement is computed as

$$\Delta\mathbf{p}_t \doteq \mathbf{p}_t^{\mathrm{odom}} - \mathbf{p}_{t-1}^{\mathrm{odom}}, \qquad \mathbf{p}_t \leftarrow \mathbf{p}_{t-1} + \Delta\mathbf{p}_t\,, \tag{4.15}$$

which is equivalent to a first-order (Euler) integration of the motion increments implicit in the odometry stream. Optionally, a fixed axis remapping/sign convention may be applied

[4] In the POSEIDON NAV stack, the `robot_state_node` publishes the fused dead-reckoning estimate on `/Odometry` (by integrating the available proprioceptive streams, e.g., DVL, IMU, and depth). In the current PoseidonSLAM runtime, this `/Odometry` topic is the sole motion input consumed by the hippocampal back-end: it drives path integration in the Pose Cell Network and provides the incremental displacements used to update the 3D Experience Map.

to $\Delta\mathbf{p}_t$ to enforce consistency with the vehicle frame conventions used by the Pose Cell and Experience Map modules.

Pose Cells

PoseidonSLAM maintains a Continuous Attractor Neural Network of Pose Cells over a discretized 4D lattice representing (x, y, z, ψ), so that the network state forms a localized "activity bump" encoding the current pose hypothesis. In the current ROS implementation, recurrent stabilization is realized by *local Gaussian excitation* combined with *global inhibition* and subsequent normalization, yielding the same functional attractor behavior typically associated with Mexican-hat/DoG dynamics in RatSLAM formulations, i.e., strong reinforcement of nearby hypotheses and suppression of competing ones.

PoseidonSLAM differs from DolphinSLAM in *how external sensory evidence drives and stabilizes the attractor*. In DolphinSLAM, the Local View layer is produced by an appearance-based BoW/FAB-MAP pipeline and, at each time step, a *single* Local View Cell is active with unit activation (l_i=1); this activation injects energy into the Pose Cell network through learned synaptic connections (Hebbian β) after a maturity period L_{mat}.

In PoseidonSLAM, by contrast, the LVC stream is produced by the neocortical HTM front-end (SP/TM), so the external drive is *graded and sequence-aware* (via TM winner-cell pooling and re-sparsification), and the implementation supports multiple simultaneously active LVC hypotheses (`multiple_local_view_active=true`). This distinction is particularly relevant in Underwater SLAM: DolphinSLAM's motion update is computed from proprioceptive sensing (DVL velocity + IMU orientation) and fused as relative motion between consecutive poses, whereas PoseidonSLAM's sequence-aware sensory correction and downstream event gating (VPR Guard) reduce the likelihood of spurious attractor pulls under perceptual aliasing before the Experience Map is updated.

Let $P_{i,j,k,q}$ denote the Pose Cell activity at lattice indices (i, j, k, q), with (i, j, k) indexing translation and q indexing yaw. Recurrent excitation is implemented as a separable Gaussian smoothing of the activity packet:

$$E_{i',j',k',q'} \;=\; \sum_{i,j,k,q} P_{i,j,k,q}\, G_{\sigma_p}(i'-i,\; j'-j,\; k'-k)\, G_{\sigma_\psi}(q'-q)\,, \tag{4.16}$$

where G_{σ_p} and G_{σ_ψ} are Gaussian kernels with variances σ_p^2 and σ_ψ^2, respectively. The excited activity is then subjected to global inhibition and renormalization:

$$P_{i',j',k',q'} \leftarrow \mathcal{N}\big(\max\big(0,\; E_{i',j',k',q'} - \phi\big)\big)\;, \tag{4.17}$$

where ϕ is a global inhibition level (uniform across the lattice), $\max(0, \cdot)$ enforces non-negativity, and $\mathcal{N}(\cdot)$ denotes a normalization operator that bounds total activity.

The recurrent Pose Cell dynamics follow the standard RatSLAM/DolphinSLAM attractor update: lateral excitation accumulates support from neighboring hypotheses, while a global inhibitory term suppresses spurious growth. In compact form, the recurrent con-

tribution at lattice index (i', j', k', q') can be written as

$$\Delta P^{\text{rec}}_{i',j',k',q'} \;=\; \sum_{i,j,k,q} P_{i,j,k,q}\, \varepsilon_{i'-i,\; j'-j,\; k'-k,\; q'-q} \;-\; \phi\,, \tag{4.18}$$

where $\varepsilon_{\Delta i,\Delta j,\Delta k,\Delta q}$ denotes the recurrent interaction kernel (excitatory neighborhood weights) and ϕ is a uniform global inhibition level.

PoseidonSLAM incorporates corrective sensory evidence through an injection mechanism that is functionally analogous to the RatSLAM injection term (Eq. (3.3)) and is learned via a Hebbian association rule as in DolphinSLAM (cf. Eq. (3.15)–(3.16)). Specifically, the synaptic coupling between an LVC identity ℓ and a Pose Cell unit (i, j, k, q) is reinforced when both are concurrently active:

$$\beta_{\ell,i,j,k,q} \leftarrow \max(\beta_{\ell,i,j,k,q},\; \lambda\, V_\ell\, P_{i,j,k,q})\,, \qquad \Delta P^{\text{inj}}_{i,j,k,q} \;=\; \delta \sum_{\ell\in\mathcal{L}} \beta_{\ell,i,j,k,q}\, V_\ell\,, \tag{4.19}$$

where $V_\ell \in \{0, 1\}$ indicates whether identity ℓ is present in the current LVC evidence (e.g., the most-active cell, or a sparse set of active cells under multi-hypothesis operation), λ is the learning rate, and δ is the injection gain. This formulation is a direct 4D extension of DolphinSLAM's Hebbian coupling and injection terms, while preserving RatSLAM's functional principle: perceptual evidence selectively amplifies pose hypotheses consistent with the recognized place identity.

Experience Map and Loop Closure

Each accepted `/experience_event` *always* instantiates a new experience node in the Experience Map. Hence, revisitation is not implemented as "node reuse" at creation time; instead, it is handled *downstream* by a matching-and-correction procedure that introduces loop closure constraints and drives map relaxation. Upon detecting a loop closure, PoseidonSLAM follows RatSLAM's relaxation philosophy (cf. Eq. (3.7)), extended to a 3D experience-graph embedding.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the 3D experience graph. Each node $i \in \mathcal{V}$ stores a 3D embedding position $\mathbf{p}_i \in \mathbb{R}^3$ and the associated place identity ℓ_i (LVC ID). Each directed edge $(i, j) \in \mathcal{E}$ stores the relative dead-reckoning displacement $\Delta\mathbf{p}_{ij} \in \mathbb{R}^3$ between consecutive events and a confidence weight $w_{ij} > 0$.

From an optimization viewpoint, the map update seeks to reduce edge residuals of the form $\mathbf{p}_j - \mathbf{p}_i - \Delta\mathbf{p}_{ij}$, which can be written as the weighted least-squares objective:

$$\min_{\{\mathbf{p}_i\}} \sum_{(i,j)\in\mathcal{E}} w_{ij}\; \|\mathbf{p}_j - \mathbf{p}_i - \Delta\mathbf{p}_{ij}\|_2^2\,. \tag{4.20}$$

A lightweight online approximation is obtained via iterative relaxation, i.e., local error redistribution over neighboring constraints, which is the direct 3D counterpart of RatSLAM's (cf. Eq. (3.7)):

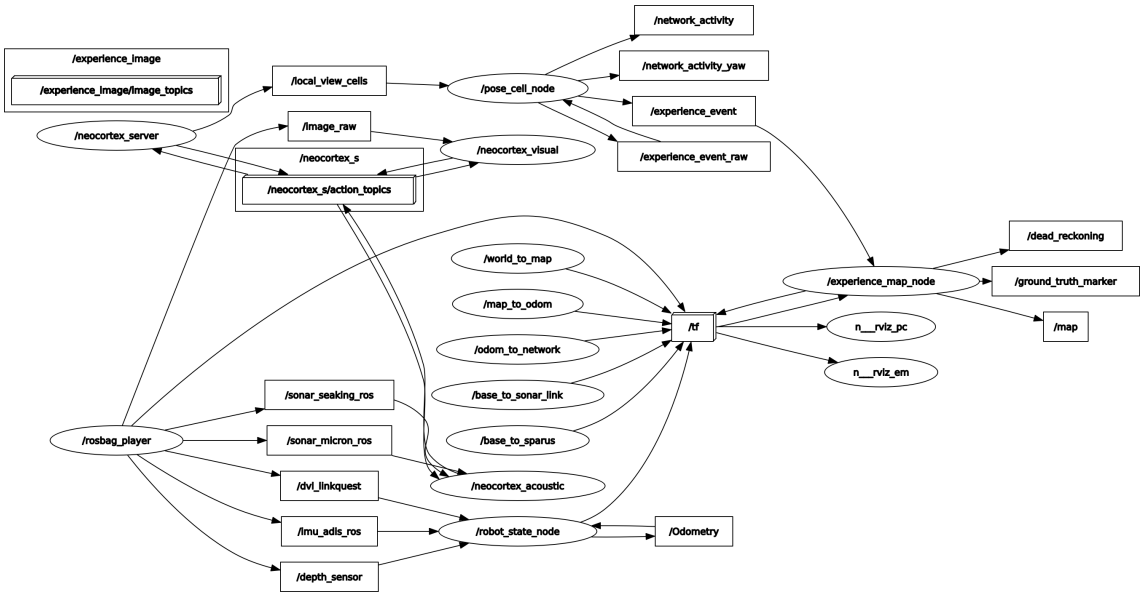


Figure 4.4: PoseidonSLAM overview: Sequence-aware place identities (LVC IDs) drive 3D experience-based mapping supported by fused dead-reckoning and loop closure correction.

$$\Delta \mathbf{p}_i \;=\; \alpha \left[\sum_{j \in \mathcal{N}_f(i)} w_{ij} \left(\mathbf{p}_j - \mathbf{p}_i - \Delta \mathbf{p}_{ij} \right) + \sum_{k \in \mathcal{N}_t(i)} w_{ki} \left(\mathbf{p}_k - \mathbf{p}_i + \Delta \mathbf{p}_{ki} \right) \right], \quad \mathbf{p}_i \leftarrow \mathbf{p}_i + \Delta \mathbf{p}_i \,, \tag{4.21}$$

where $\mathcal{N}_f(i)$ and $\mathcal{N}_t(i)$ denote the forward and backward neighbors of node i, respectively.

In the current ROS implementation, loop closure corrections are applied as bounded updates to mitigate topological collapse under spurious recognitions. Concretely, the correction magnitude is distributed along the inferred route length (yielding an effective route-dependent step size) and is further constrained by safety clamps such as `max_correction_norm=20.0` and `max_correction_step=0.05`, which cap the overall correction norm and the per-iteration update, respectively.

4.2.5 ROS Implementation

PoseidonSLAM is implemented as a modular, ROS-native system (Fig. 4.4) designed for repeatable, end-to-end execution under dataset playback. The *neocortical front-end* is deployed as an action server with dedicated sensor clients (visual and acoustic) that produce a continuous stream of `/local_view_cells`. The hippocampal back-end is realized by three nodes: (i) a *robot state node* that publishes the fused dead-reckoning stream `/Odometry` (Eq. (4.15)); (ii) a *pose cell node* that maintains the 4D CANN dynamics and applies the event-consistency logic (Eq. (4.11)); and (iii) an *experience map node* that constructs and relaxes the 3D experience graph (Eqs. (4.20)–(4.21)).

Interfaces

The principal ROS interfaces are:

- **Inputs:** visual stream (`/camera/image_raw`); acoustic streams (`/sonar_micron_ros`, `/sonar_seaking_ros`); proprioception and navigation sensors used by the robot state module (`/dvl_linkquest`, `/imu_adis_ros`, `/depth_sensor`); and dataset odometry used as an external reference when available (`/odometry`).
- **Front-end output:** Local View Cells (`/local_view_cells`).
- **Back-end outputs:** Fused dead-reckoning (`/Odometry`), experience events (`/experience_event`), topological experience graph (`/map`), dead-reckoning trace (`/dead_reckoning`), and event evidence (`/experience_image`). When provided by the dataset, `/odometry` is additionally logged as a reference signal (e.g., EKF-derived) and published as `/ground_truth`.

Event–Evidence Audit Contract

For experimental traceability, PoseidonSLAM publishes an auditable sensory snapshot for each accepted experience. Concretely, the experience-map node publishes `/experience_image` as the closest camera frame to the `/experience_event` timestamp within a configurable tolerance `sync_stale_sec`. The header of `/experience_image` (sequence and timestamp) is set equal to the corresponding `/experience_event` header, enabling deterministic event–evidence joins during offline analysis.

Operational Flow

Algorithm 1 PoseidonSLAM operational flow.

Initialize nodes: neocortical action server and sensor clients; robot-state (fused DR); pose cells; experience map.
for each time step t with sensor observations **do**
    encode visual+acoustic observations → modality SDRs (Eqs. (4.1)–(4.2))
    fuse SDRs by union → $\mathbf{s}_t^{\text{fused}}$ (Eq. (4.3))
    SP/TM processing → sequence-aware matching descriptor and LVC identity
    publish `/local_view_cells` (continuous perceptual stream)
    update dead-reckoning and pose cell dynamics using `/Odometry` (Eq. (4.15))
    if VPR Guard accepts the identity stream (Eq. (4.11)) **then**
        publish `/experience_event`
        update experience graph; correct upon loop closures (Eqs. (4.20)–(4.21))
        publish `/map`, `/dead_reckoning`, and `/experience_image`

Configuration

PoseidonSLAM parameters are loaded at launch time (ROS parameter server), with the `.launch` file taking precedence over YAML defaults when both are provided. Representative parameters include:

- **Front-end:** SDR dimension L and sparsity control (Eq. (4.2)), union-based fusion (Eq. (4.3)), interval consolidation ($K_{\max}$ in Eq. (4.5)), and LVC matching thresholds (Eq. (4.9)).
- **Back-end:** 4D lattice sizes (x, y, z, ψ) and recurrent dynamics parameters (Eq. (4.18)), LVC→PC coupling gains (Eq. (4.19)), relaxation and safety clamps in the experience graph (Eq. (4.21)), and the VPR Guard filter $(w, r, L_{\max})$ (Eq. (4.11)).

For reproducibility, the random projection matrix is fixed and persisted (seed 42 if regeneration is required), and each run stores a full parameter snapshot (e.g., `rosparam dump`), together with the bag identifier and the code revision hash used for execution.

Table 4.1 consolidates the **default** PoseidonSLAM settings as defined by `poseidon_nav.launch`. The launch file loads `poseidon_nav.yaml` and then applies explicit overrides; therefore, the values below correspond to the **launch defaults** (i.e., fixed `<param ...>` values and `<arg ... default=...>` values).

Table 4.1: PoseidonSLAM default configuration: blocks, parameters, and settings (ROS Melodic). Values correspond to the default `poseidon_nav.launch` configuration.

Parameter	Description and default setting
Global synchronization and audit	
`sync_stale_sec`	Canonical time window (seconds) used consistently across PoseidonSLAM to bound cross-topic associations (e.g., neocortex sensor-to-goal alignment and ExperienceMap event–image auditing). **Default:** `0.75`.
Front-end: fusion, CNN and LSBH	
`fusion_mode`	**Effective fusion operator** used to combine per-modality SDRs into a single fused SDR before SP/TM. Fusion is a binary union (bitwise OR), followed by sparsity enforcement to `sp_numActiveColumns`. **Default (launch):** `union`.
`cnn_backbone`	Pre-trained CNN backbone used as the feature extractor for the perceptual encoder (image stream and sonar-derived polar tensors). **Default:** `shufflenetv2`.

Continues on next page

Table 4.1 (continued)

Item	Description and default setting
`encoder_mode`	Per-modality encoder used to generate SDRs. **Default:** `lsbh`.
`lsbh_matrix_path`	LSBH projection matrix. **Default:** `$(find neocortex)/src/lsbhMatrixShuffleNetV2.npy`.
`lsbh_s`	LSBH selection ratio (top/bottom fraction). **Default:** `0.25`.
`rp_matrix_path`	Random projection matrix (only for non-default RP mode). **Default:** `$(find neocortex)/src/randomMatrixShuffleNetV2.npy`.
Front-end: HTM Spatial Pooler (neocortex_ server)	
`sp_enable`	Enable Spatial Pooler. **Default:** `true`.
`sp_columns`	SDR dimensionality (number of bits) used throughout the front-end representation. **Default:** `2048`.
`sp_numActiveColumns`	Target sparsity (number of active bits kept after fusion/binarization and before SP/TM updates). **Default:** `20` (i.e., $\approx 0.98\%$ of 2048).
`sp_potentialRadius`	Spatial Pooler potential radius. **Default:** `600`.
`sp_synPermActiveInc`, `sp_synPermInactiveDec`	SP permanence update rates. **Default:** `0.006`, `0.008`.
Front-end: HTM Temporal Memory and sequence descriptor	
`tm_cellsPerColumn`	Number of TM cells per SP column (controls TM state size and sequence capacity). **Default:** `40`.
`tm_activationThreshold`	TM activation threshold. **Default:** `5`.
`tm_permanenceIncrement`, `tm_permanenceDecrement`	TM permanence update rates. **Default:** `0.008`, `0.012`.
`sv_descriptor_source`	Descriptor type used to represent the local view identity for SV-cell matching. **Default:** `tm`.
`sv_tm_window`	History length (in steps) used to aggregate TM winner cells before matching. **Default:** `7`.
`sv_tm_topk`	Top-k cap applied to the TM-derived descriptor (most frequent winners across the window). **Default:** `15`.
`sv_tm_match_threshold`	Primary acceptance threshold for SV matching when `sv_descriptor_source=tm`. **Default:** `0.30`.
Front-end: LVC ID and matching gates	

Continues on next page

Table 4.1 (continued)

Item	Description and default setting
`interval_mode`	If enabled, applies NeoSLAM-like interval union for short-run stabilization. **Default:** `true`.
`theta_alpha`, `theta_rho`	Interval union parameters. **Default:** `0.65`, `3`.
`sv_match_threshold`	Base acceptance threshold for SV matching (overlap score $\|A \cap B\|/\|A\|$). **Default:** `0.38`.
`sv_min_intersection`	Minimum absolute intersection size. **Default:** `12`.
`sv_require_multimodal`	Require per-modality validation for SV match. **Default:** `false`.
`sv_min_img_score` / `sv_min_sonar_score`	Per-modality overlap thresholds (only active when `sv_require_multimodal=true`). **Default:** `0.45` / `0.35`.
`sv_enable_f1_gate`, `sv_min_f1`	Optional F1 gate for descriptor densification. **Default:** `true`, `0.55`.
`sv_use_iou_tiebreak`, `sv_max_bits`	Tie-break by IoU and max-bit cap. **Default:** `true`, `0`.
Front-end: sonar preprocessing (neocortex_server)	
`sonar_polar_width` × `sonar_polar_height`	Output polar tensor resolution used to rasterize sonar scans before CNN encoding. **Default:** `224`×`224`.
`sonar_rmax`, `sonar_use_intensities`	Maximum sonar range (meters) and whether intensity values are used. **Default:** `30.0`, `true`.
Back-end: CANN and pose cells	
`/cann/number_of_neurons`	CANN lattice resolution per latent dimension (4D). **Default:** `[15,15,15,8]`.
`/cann/distance_between_neurons`	Inter-neuron spacing for translational dimensions (meters). **Default:** `[0.1,0.1,0.1]`.
`pose_cells.top_k`, `pose_cells.min_exc`	Pose-cell selection parameters. **Default:** `4`, `0.03`.
Back-end: VPR Guard, motion gate, and frame conventions (pose_cell_node)	
`require_vpr_for_event`	If true, experience events are emitted only when the VPR Guard accepts the current identity. **Default:** `true`.
`vpr_window`, `vpr_min_rep`	VPR Guard window size and minimum repetitions required. **Default:** `26`, `3`.
`vpr_leak_rate`	Leak parameter (consecutive rejections). **Default:** `20`.

Continues on next page

Table 4.1 (continued)

Item	Description and default setting
`motion_gate_enable`	Requires a minimum displacement before creating an experience event. **Default:** `true`.
`min_event_trans_xy`, `min_event_dz`	Minimum translational displacement thresholds (meters). **Default:** `0.12`, `0.05`.
`axis_remap`, `axis_sign_z`	Dead-reckoning frame alignment convention. **Default:** `yxz`, `-1.0`.
Back-end: Experience Map (experience_map_node)	
`match_threshold`	Loop-closure matching acceptance threshold (each event still creates a new node). **Default:** `0.98`.
`lv_factor`, `pc_factor`	Relative weighting between local-view evidence and pose-cell evidence. **Default:** `0.5`, `0.5`.
`min_experience_age`, `loop_confirmations`	Minimum node age before loop-closure confirmation and number of confirmations required. **Default:** `200`, `2`.
`min_correction_route_len`	Minimum route length required for correction. **Default:** `250`.
`max_correction_norm`, `max_correction_step`	Safety clamps on correction magnitude and per-step update. **Default:** `20.0`, `0.05`.
`min_correction_interval_sec`	Minimum time between correction applications. **Default:** `15.0`.
Audit (image): event–evidence traceability (experience_map_node)	
`image_topic`, `image_transport`	Camera topic and transport used to fetch the source images for experience evidence. **Default:** `/camera/image_raw`, `raw`.
`publish_experience_image`	If enabled, publishes `/experience_image` for post-run auditing. **Default:** `true`.
`image_buffer_size`, `image_sync_slop`	Buffer size (images) and association slop (seconds). **Default:** `400`, `0.75` (tied to `sync_stale_sec`).

4.3 Contributions

PoseidonSLAM provides:

- **Neuro-inspired Visual-Inertial-Acoustic Underwater SLAM** integration coupling sequence-aware multimodal place identities to 3D experience-graph mapping, explicitly grounded on RatSLAM back-end principles;

- **Multimodal Sparse Distributed Representations** based on fixed random projections, Top-k sparsification, and union fusion for robustness under sensing degradation and dropouts;
- **Sequence-aware LVC identities** derived from HTM Temporal Memory context (winner-cell dynamics), explicitly distinct from RatSLAM SAD templates and DolphinSLAM FAB-MAP probabilistic activations;
- **4D Pose Cell Network** over (x, y, z, ψ) with local Gaussian excitation and global inhibition, using an LVC→PC association analogous to DolphinSLAM and functionally equivalent to RatSLAM injection;
- **3D Experience Map** construction and loop closure correction as a direct 3D extension of RatSLAM's experience-map optimization, with safety clamps to mitigate collapse under spurious closures;
- **Reproducible ROS implementation** with deterministic event–evidence auditing (Section 4.2.5) and run-time parameter/version logging for traceable experimentation.

Chapter 5

Experimental Validation and Results

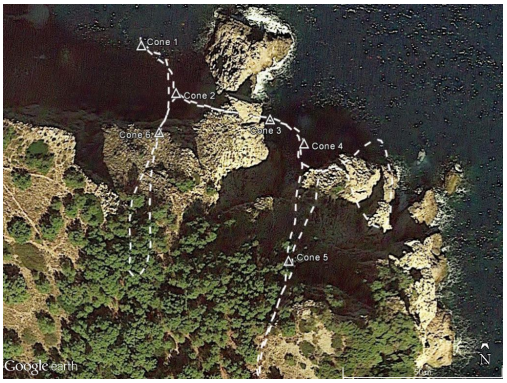
Having established the theoretical foundations, this chapter presents the empirical validation of the proposed neocortical-inspired Visual SLAM framework and its instantiation within the *POSEIDON NAV* pipeline (**PoseidonSLAM**). The primary goal is to demonstrate perceptual robustness and map consistency in a challenging (simulated-from-real) underwater setting where state-of-the-art methods from both geometric and canonical bio-inspired paradigms fail. We detail the experimental design, the creation of a benchmark dataset, the implementation of all systems, and a comparative analysis that supports the central contributions of this dissertation. In addition, we report an ablation study (**VPR Guard ON vs. OFF**) that audits the traceability between topological events and their perceptual evidence and characterizes the resulting **3D topological map stability** (proxy metrics; not a substitute for full localization evaluation).

5.1 Benchmark Dataset: The Underwater Cave Experiment

The original experiment was conducted by MALLIOS *et al.* [5] in the Coves de Cala Viuda, a complex and unstructured underwater cave system in Spain (Figure 5.1). For the experiment, the Sparus AUV—a compact, hovering-capable vehicle—was tasked with mapping the cave system. To ensure the collection of a high-quality dataset suitable for benchmarking SLAM algorithms, the AUV was guided by a diver along an approximately 500 m trajectory explicitly designed to include multiple loop closures [6]. The vehicle was equipped with a comprehensive sensor suite, including a horizontally-mounted Mechanical Scanning Imaging Sonar (MSIS) as the primary perception sensor for a scan-matching SLAM algorithm, alongside a DVL and IMUs for dead-reckoning [5].

Of particular relevance to this dissertation is the visual-inertial sensor package used. The visual data was captured by a low-cost, downward-facing analog PAL camera at a resolution of 384×288 pixels and a frame rate of approximately 4 Hz. For inertial measurements, the platform included two IMUs, of which the Analog Devices ADIS16480 was mounted externally to minimize electromagnetic interference, providing a high-quality data

(a) Coves de Cala Viuda.



(b) Arrangement of ground truth beacons.

Figure 5.1: The environment from the original Underwater Cave Experiment. Source: Reproduced from [5].

stream for visual-inertial fusion [6].

A key aspect of the original work, and a primary motivator for our research, was the challenge posed by the visual conditions. MALLIOS *et al.* [5] report that, due to turbidity, data from the AUV's video systems could not be used for comparison with visual-odometry or visual-SLAM techniques. The visual stream was therefore used only to identify pre-deployed traffic cones for ground truth validation (Figure 5.2). This foundational experiment thus provides not only the realistic visual data upon which our simulation is built but also a clear benchmark of the conditions under which traditional perception methods fail.

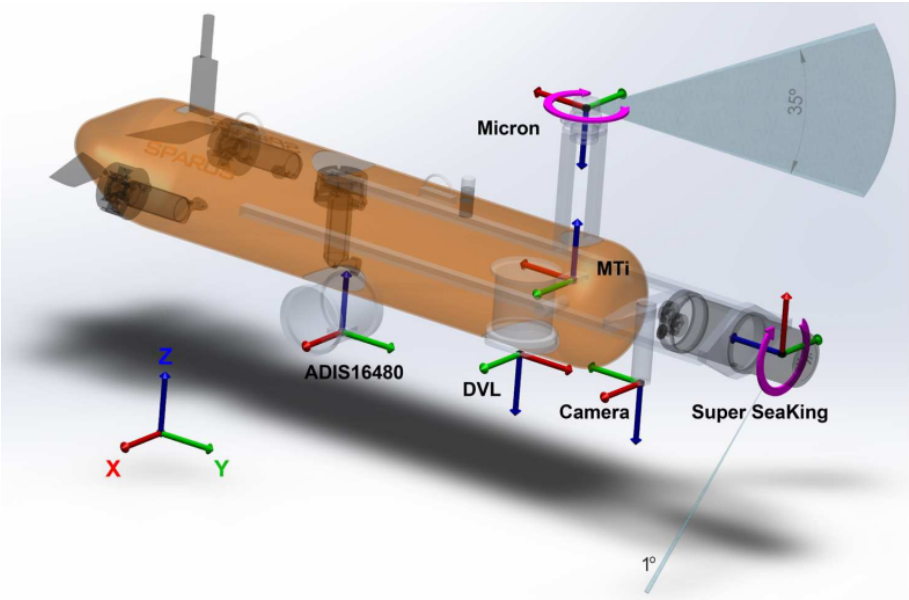


Figure 5.2: AUV Sparus II used on the Underwater Cave Experiment. Source: Reproduced from [5].

The full dataset is publicly available and was distributed by its authors as two separate

ROS bag files[1]: `full_dataset.bag` (395 MB), containing all sensor data except the camera, and `sparus_camera.bag` (3.3 GB), containing the visual data stream. For the purposes of this dissertation, these two files were merged into a single, chronologically-ordered *rosbag* named `merged_sparus_data.bag`. The final merged dataset has a duration of 32 minutes and 35 seconds and comprises a total of 690,649 messages. This comprehensive dataset provided all the necessary inputs for our experiments, including 9,774 raw image messages (`/camera/image_raw`), 19,965 inertial messages from the high-fidelity ADIS16480 IMU (`/imu_adis_ros`), and 258,905 transform messages (`/tf`) that define the vehicle's static sensor extrinsic calibrations, enabling a meticulous and repeatable setup for the ORB-SLAM3 baseline and the proposed neocortical-inspired framework.

5.2 The Underwater Cave Transformed Database

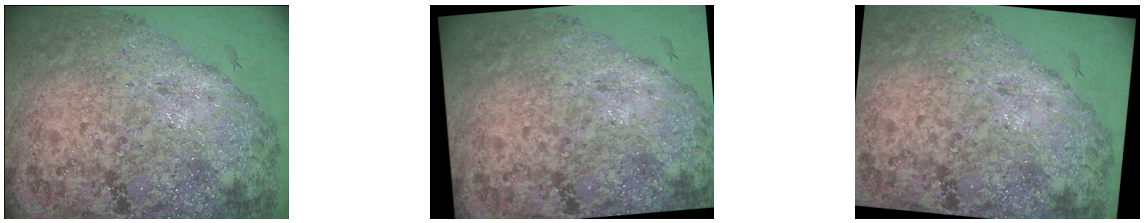
To specifically test long-term robustness and resilience to subtle visual changes during repeated traversals, we created the **Underwater Cave Transformed Database**. This involved a process of re-structuring the original dataset to simulate a longer, more challenging trajectory. Using custom scripts, segments of the original data were concatenated to create a new path where the AUV passes through two distinct areas multiple times. The resulting trajectory contains nine sequential segments, including three consecutive loops in a first area ("Ellipse") followed by three consecutive loops in a second area ("Cave 2"), connected by transition paths.

The core of the innovation lies in comprehensive data synthesis for repeated loops. While the first loop in each area uses the original, unaltered sensor data, the second and third loops were modified to simulate challenges of long-term autonomy:

- **Visual Transformation:** Each image in the second and third loops was programmatically altered, subjected to random horizontal translations (mean of 5%±1% of image width) and rotations (mean of 0°±0.5°). This process, illustrated in Figures 5.3 and 5.4, creates a scenario where the system must recognize previously visited locations despite viewpoint drift.
- **Comprehensive Sensor Data Synthesis:** The full suite of sensor readings from the first loop of each area—including proprioceptive data from the **IMU** and **DVL**, and exteroceptive data from the **depth sensor** and the two imaging **sonars**—was replicated and synchronized with the odometry of the subsequent loops. This ensures that all sensor streams correspond consistently to the repeated path, producing a rich dataset suitable for evaluating multimodal fusion architectures such as *POSEIDON NAV*.

The resulting synthetic benchmark is packaged as a single ROS bag file: `underwater_-`

[1] https://cirs.udg.edu/caves-dataset/



(a) First pass (b) Second pass (c) Third pass

Figure 5.3: Examples of the programmed visual transformations applied to create the *Underwater Cave Transformed Database*. The first column shows original images (Looping Area 1). The second and third columns show subsequent passes after applying random translations and rotations. This simulates viewpoint variations inherent in multi-session navigation. Source: Author's own creation based on [5].

(a) First pass (b) Second pass (c) Third pass

Figure 5.4: Examples of the programmed visual transformations applied to create the *Underwater Cave Transformed Database*. The first column shows original images (Looping Area 2). The second and third columns show subsequent passes after applying random translations and rotations. Source: Author's own creation based on [5].

`cave_transformed_database.bag`[2]. This 6.4 GB dataset spans a simulated trajectory of 1 hour, 0 minutes, and 22 seconds (3,622 s) and comprises 557,185 messages. It provides a comprehensive suite of synchronized streams structured for SLAM evaluation, including: (i) 18,115 visual frames (`/image_raw`, `/image_compressed`); (ii) 36,998 IMU and 36,998 odometry messages; (iii) 97,473 SeaKing and 45,586 Micron sonar scans; (iv) 19,548 depth readings and 5,562 DVL messages; and (v) 258,831 `/tf` messages.

5.3 System Implementation and Evaluation Metrics

5.3.1 ORB-SLAM3 System Configuration

The experimental validation pipeline was implemented within the Robot Operating System (ROS) framework. To establish a state-of-the-art geometric baseline, **ORB-SLAM3** was executed in Monocular-Inertial mode on the Transformed Database. The system was configured using `sparus_final_calib.yaml`, following published dataset specifications. Intrinsics and distortion parameters matched the calibration contained in the ROS bags. The extrinsic transform between camera and the ADIS16480 IMU (`Tbc`) was configured based on offsets reported by MALLIOS *et al.* [6]. The IMU noise model parameters were set to typical MEMS values to satisfy the tightly coupled MAP optimization assumptions [4]. To maximize feature detection in underwater imagery, ORB extractor parameters were tuned by increasing the number of features and lowering FAST thresholds.

This configuration included:

- **Camera Model:** Pin-hole intrinsics and distortion parameters consistent with the dataset, at **384×288** and **4 Hz**.
- **Visual-Inertial Setup:** `Tbc` from MALLIOS *et al.* [6]; IMU noise parameters (NoiseGyro: 1.7e-4, NoiseAcc: 2.0e-3, GyroWalk: 1.9e-5, AccWalk: 3.0e-3) aligned with sensor specifications [4].
- **Feature Extraction Adaptation:** `nFeatures`=3200, `iniThFAST`=5, `minThFAST`=3 to increase the probability of detecting stable corners in low-contrast scenes.

5.3.2 NeoSLAM System Configuration

The **NeoSLAM** system was implemented in ROS as detailed in Chapter 3. Parameters in Table 5.1 were configured based on Hierarchical Temporal Memory (HTM) principles [70]. In the Spatial Pooler, `columnDimensions`=2048 sets the SDR dimensionality (n), and `numActiveColumnsPerInhArea`=20 fixes the sparsity ($w = 20$) at approximately 1%. In the Temporal Memory, `cellsPerColumn`=40 enables high-order sequence representations; `activationThreshold`=5 supports predictive states under partial evidence; and permanence dynamics were chosen for stable gradual learning [70].

[2] https://drive.google.com/drive/folders/1jg4BppsiaAlIWiiR8c6KY_v8jA6f-zsI?usp=sharing

Table 5.1: NeoSLAM algorithm configurations for the *Underwater Cave Transformed Database*, with parameter descriptions grounded in HTM theory.

Spatial Pooler Configuration		
Parameter	**Value**	**Description**
inputDimensions	2048	Dimensionality of the input SDR from the CNN encoder.
columnDimensions	2048	Dimensionality (n) of the output SDR; total number of columns.
potentialRadius	600	Receptive field for each column's potential synapses.
numActiveColumnsPerInhArea	20	Number of winning columns (w); controls sparsity ($\approx$1%).
globalInhibition	True	Global competition across the layer for higher performance.
Temporal Memory Configuration		
Parameter	**Value**	**Description**
columnDimensions	2048	Total number of columns in the TM layer.
cellsPerColumn	40	Cells per column; enables high-order temporal contexts.
activationThreshold	5	Active synapses required to trigger predictive state.
initialPermanence	0.55	Initial permanence for new synapses.
connectedPermanence	0.5	Permanence threshold for connected synapses.
maxNewSynapseCount	20	Max new synapses grown per step.
permanenceIncrement	0.008	Reinforcement rate for correct predictions.
permanenceDecrement	0.012	Punishment rate for incorrect predictions.

5.3.3 Precision–Recall Curve

To quantitatively evaluate loop-closure detection performance, we use standard metrics from Visual Place Recognition :

- **Precision (P):**

$$P = \frac{TP}{TP + FP} \tag{5.1}$$

- **Recall (R):**

$$R = \frac{TP}{TP + FN} \tag{5.2}$$

- **F1-Score:**

$$F1 = \frac{2 \cdot P \cdot R}{P + R} \tag{5.3}$$

5.4 Results and Analysis

This section presents comparative results against state-of-the-art baselines. We first document the failure modes of ORB-SLAM3 and RatSLAM on our underwater benchmark. We then present successful results obtained by NeoSLAM with two CNN front-ends (AlexNet-conv3 and ShuffleNetV2 x1.0). Finally, we report **PoseidonSLAM** results (NeoSLAM-derived topological mapping integrated in *POSEIDON NAV*) with a targeted ablation (**VPR Guard ON vs OFF**) focused on 3D topological map stability and event–evidence traceability.

5.4.1 ORB-SLAM3 Monocular-Inertial Failure Analysis

To establish a rigorous baseline, we evaluated ORB-SLAM3 [4] in Monocular-Inertial mode on the Transformed Database using the tuned `sparus_final_calib.yaml`. Despite these optimizations, the system failed to process the full trajectory. ORB-SLAM3 initializes and begins tracking, but experiences catastrophic tracking failure in visually degraded and repetitive sections. Figure 5.5 illustrates a representative moment where the front-end cannot extract and match a sufficient number of stable ORB features from turbid, low-contrast imagery. The system enters a "LOST" state and cannot produce a coherent map or trajectory.

This result empirically confirms the limitations of feature-based methods in perceptually challenging underwater environments. The failure mechanism aligns with the known sensitivity of ORB-SLAM3 in low-texture scenes [4]. This establishes a clear benchmark: even a geometry-based system fused with inertial data collapses due to brittleness of the visual front-end.

5.4.2 RatSLAM Visual-Inertial Failure Analysis

As a canonical bio-inspired baseline, we adopted RatSLAM as implemented in OpenRatSLAM. The system comprises a Local View module for template-based place recognition, a Pose Cell continuous-attractor network for path integration, and a topological Experience Map for global consistency. We followed established tuning procedures that prioritize the visual front-end [8]. Given the challenging underwater imagery (low texture, variable

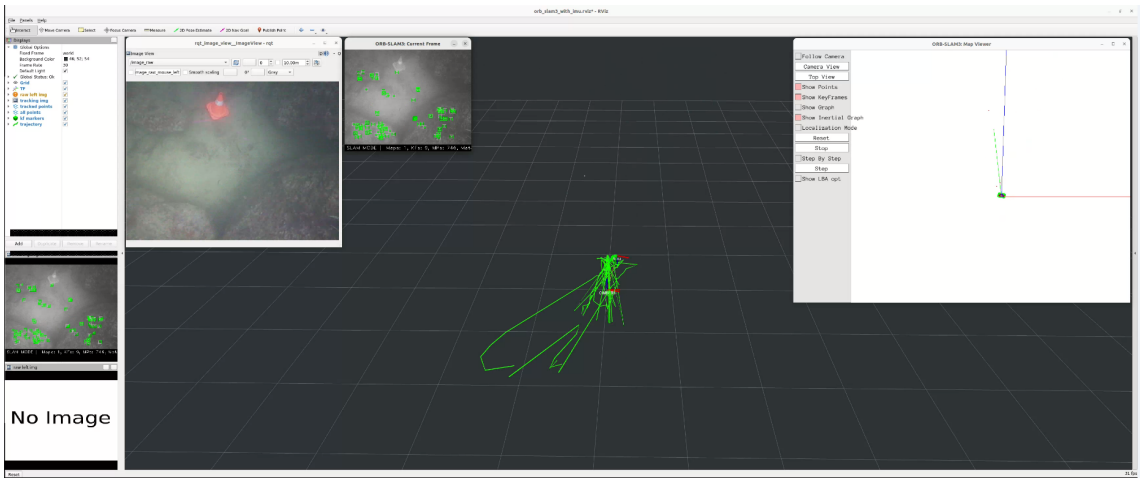


Figure 5.5: Catastrophic tracking failure of **ORB-SLAM3 Monocular-Inertial** on the *Underwater Cave Transformed Database*. The main window shows the system in a "LOST" state (left). The map viewer (right) shows an incomplete and incoherent trajectory generated before failure. Source: The author.

illumination, perceptual aliasing), our configuration `ratslam_irataus.yaml` was designed to (i) bias templates toward the seabed by cropping (`image_crop_y_min`=120); (ii) enable global and local patch normalization (`vt_normalisation`=0.4, `vt_patch_normalise`=3); (iii) enforce a restrictive matching threshold (`vt_match_threshold`=0.018); and (iv) reduce brittle relocalizations by lowering visual injection energy (`pc_vt_inject_energy`=0.14).

Under the tuning protocol adopted in this study, OpenRatSLAM still could not produce a coherent map. The Experience Map collapses into a topologically inconsistent structure due to repeated false matches between visually similar but distinct locations. This highlights that SAD-based low-resolution templates remain fundamentally fragile in this underwater setting.

5.4.3 NeoSLAM: Successful and Robust Visual Place Recognition

In contrast to the baseline failures, NeoSLAM successfully processed the full trajectory and produced a coherent topological map. Figures 5.7 and 5.8 show live outputs for AlexNet-conv3 and ShuffleNetV2 x1.0 front-ends. In both cases, the Pose Cell activity packet remains stable while an Experience Map is constructed, overcoming the tracking loss and map collapse that affected ORB-SLAM3 and RatSLAM.

Visual Similarity and Place Recognition

The first quantitative evidence of NeoSLAM's perceptual robustness is given by similarity matrices (Figures 5.9 and 5.10), which visualize pairwise similarity across frames. The AlexNet-conv3 matrix (Figure 5.9) exhibits more off-diagonal noise, indicating higher risk of perceptual aliasing. In contrast, ShuffleNetV2 (Figure 5.10) shows cleaner off-diagonals,

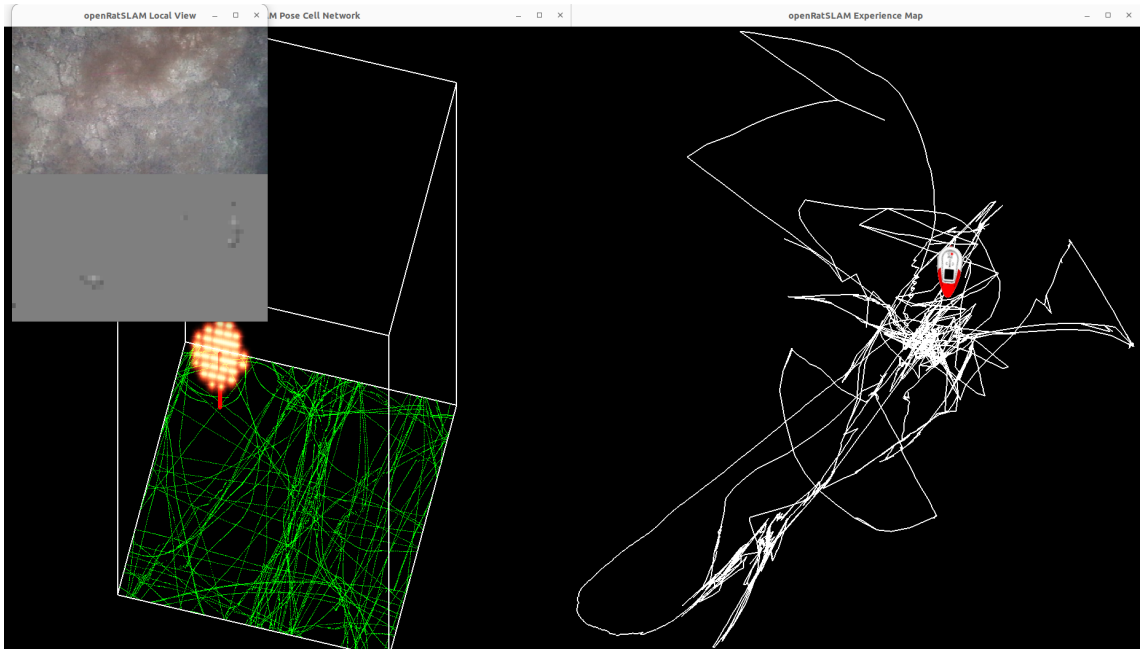


Figure 5.6: Catastrophic mapping failure of **OpenRatSLAM** on the *Underwater Cave Transformed Database*. The Experience Map (right) fails to converge, collapsing into a tangled, topologically incoherent graph. The Local View (left) shows ambiguous, low-feature imagery that drives unreliable place recognition. Source: The author.

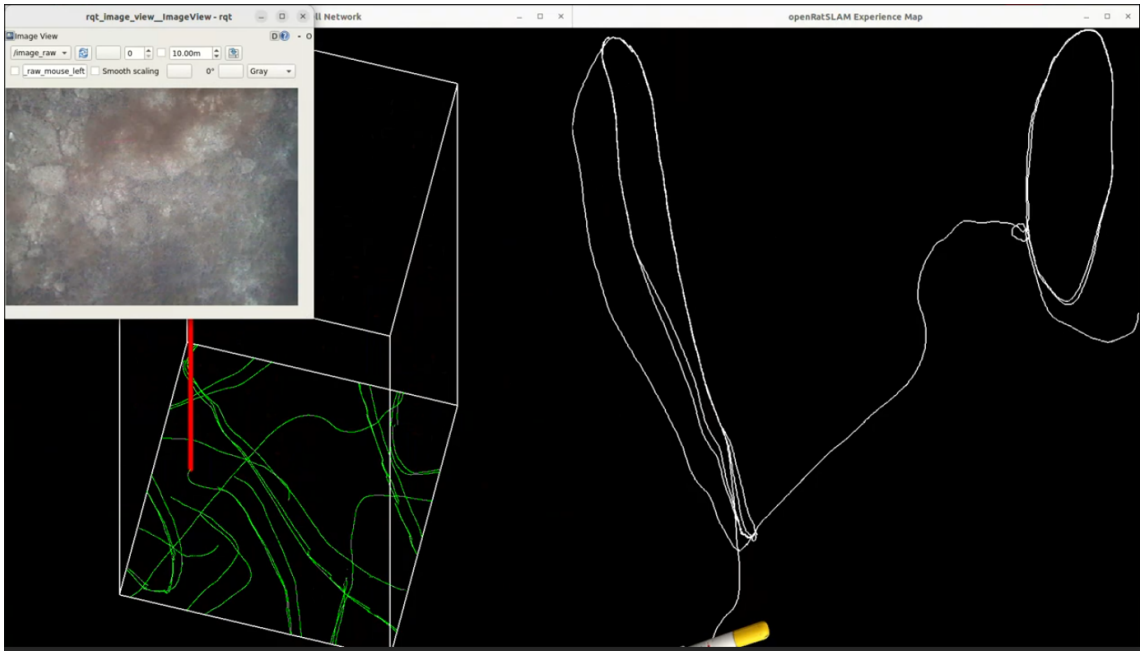


Figure 5.7: Successful execution of NeoSLAM on the *Underwater Cave Transformed Database* with the AlexNet-conv3 front-end. The system produces a coherent Experience Map by maintaining stable Pose Cell dynamics. Source: The author.

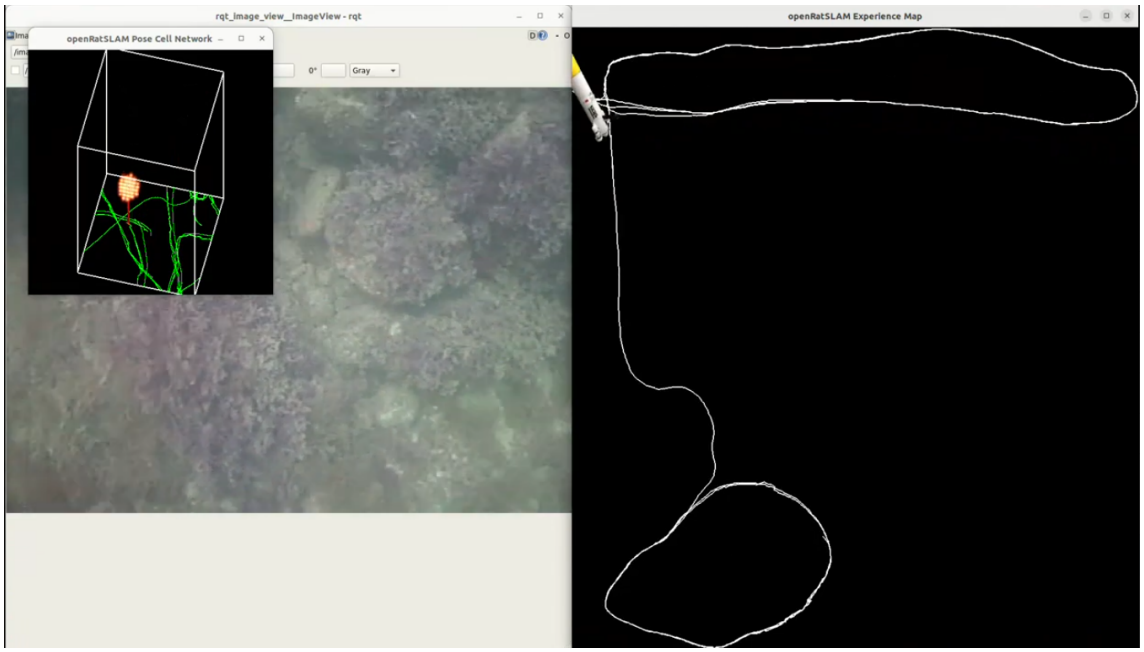


Figure 5.8: Successful execution of NeoSLAM with the ShuffleNetV2 x1.0 front-end. The resulting Experience Map is visibly more coherent in looped regions, suggesting improved perceptual discrimination. Source: The author.

supporting more selective place matches.

Landmark Creation and Loop Closure Detection

View Cell plots (Figures 5.11 and 5.12) visualize landmark creation (blue crosses) and loop closures (red circles). AlexNet-conv3 produced 2,842 view cells and detected 147 loop closures (Figure 5.11). ShuffleNetV2 produced fewer view cells (2,471) but detected substantially more loop closures (266), evidencing improved perceptual efficiency.

Trajectory Correction

Figures 5.13 and 5.14 show final trajectories. Both configurations correct significant odometry drift (blue) to produce coherent topological maps (red), but ShuffleNetV2 yields more frequent corrections due to higher loop-closure yield.

Ground Truth for Performance Evaluation

Ground truth (GT) matrices (Figures 5.15 and 5.16) define ideal loop closures based on the known, pre-programmed trajectory of the transformed dataset. They enable classification of detections into TP/FP/FN for P–R computation. Separate GT matrices are generated per experiment to match the number of view cells.

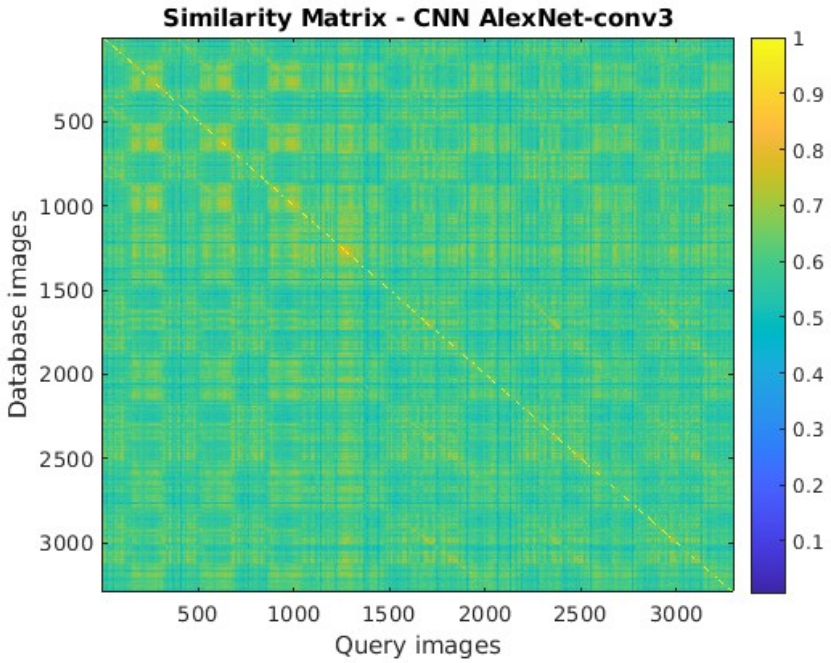


Figure 5.9: Similarity matrix for the AlexNet-conv3 front-end. Increased off-diagonal noise suggests a higher potential for ambiguous place matches. Source: The author.

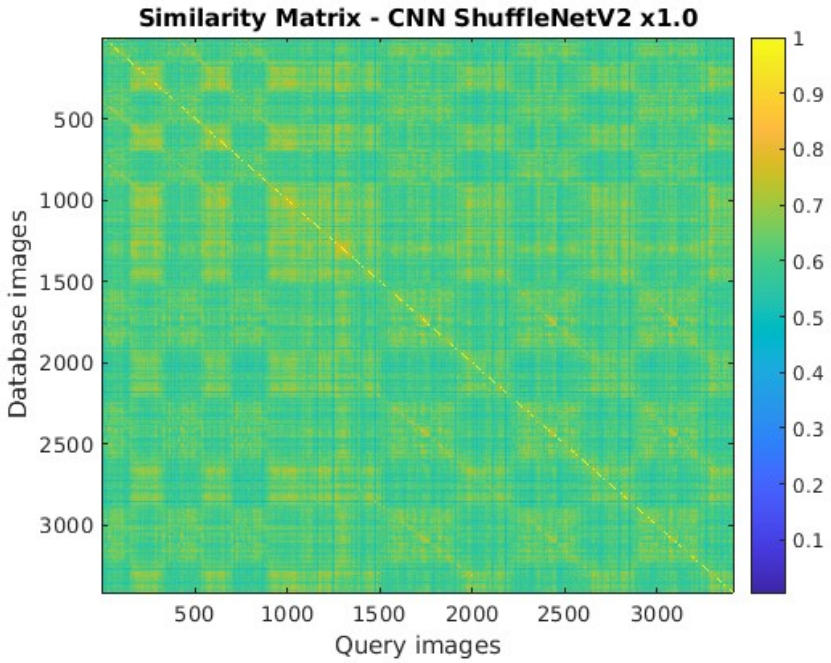


Figure 5.10: Similarity matrix for ShuffleNetV2 x1.0. Reduced off-diagonal noise relative to Figure 5.9 indicates fewer ambiguous matches and higher discriminative power. Source: The author.

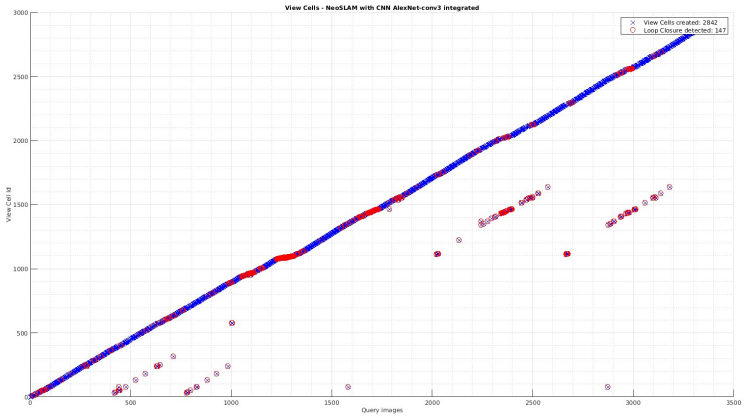


Figure 5.11: View Cells generated with AlexNet-conv3. The plot shows new templates (blue crosses) and detected loop closures (red circles). Source: The author.

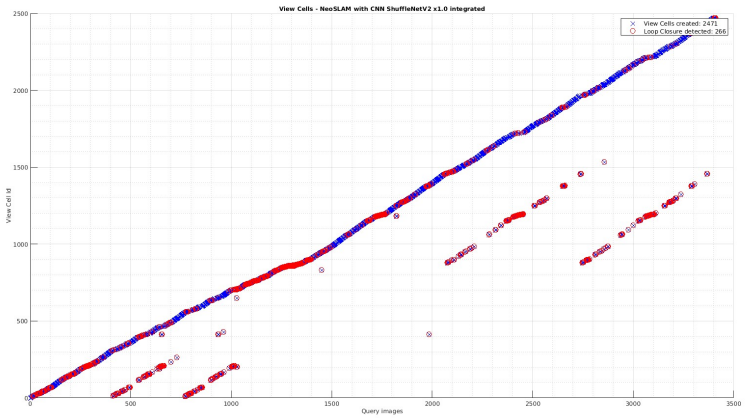


Figure 5.12: View Cells generated with ShuffleNetV2 x1.0. The higher number of loop closures from a smaller set of landmarks indicates greater perceptual efficiency. Source: The author.

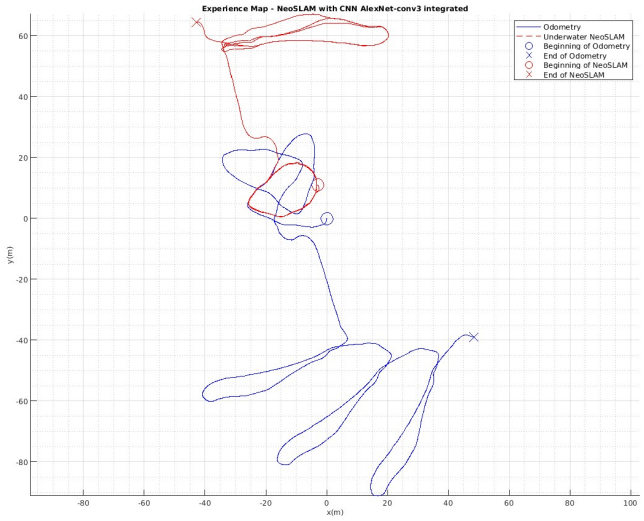


Figure 5.13: Final trajectory with AlexNet-conv3. The system corrects raw odometry drift to produce a coherent topological map. Source: The author.

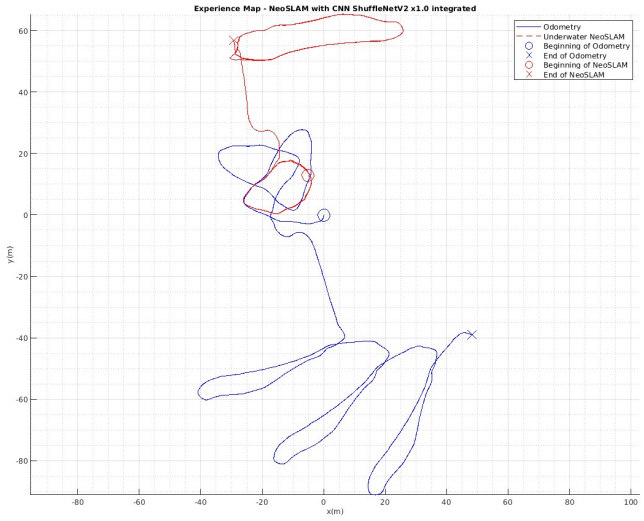


Figure 5.14: Final trajectory with ShuffleNetV2 x1.0. Increased loop closures yield visibly more stable loop geometry compared to Figure 5.13. Source: The author.

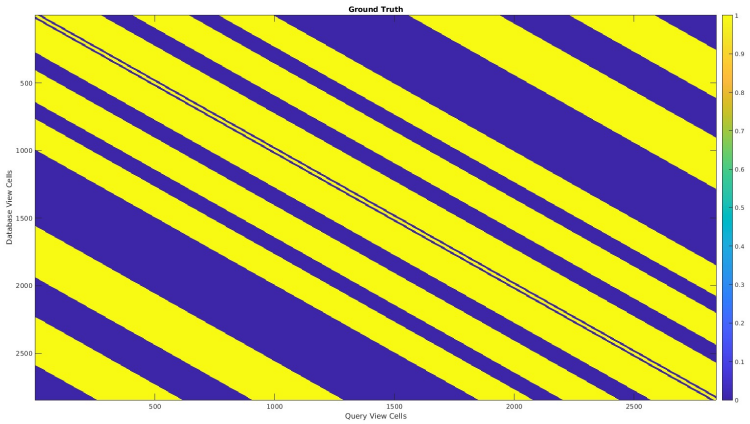


Figure 5.15: Ground truth matrix for AlexNet-conv3. Bright off-diagonal bands represent correct loop-closure opportunities based on the known trajectory. Source: The author.

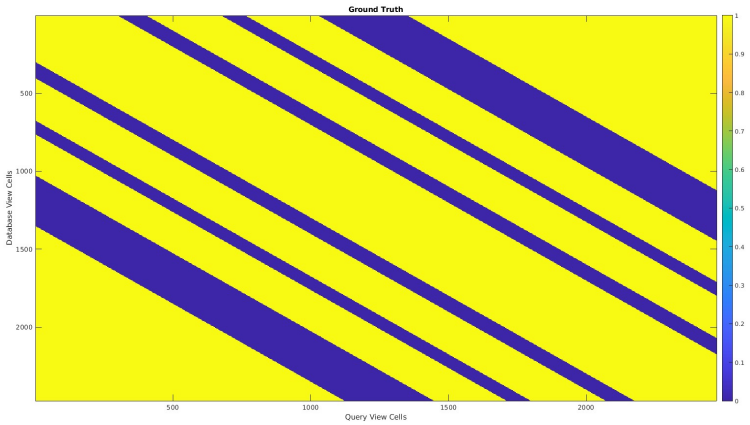


Figure 5.16: Ground truth matrix for ShuffleNetV2 x1.0, used to evaluate precision and recall of loop-closure detections. Source: The author.

Precision–Recall Analysis

Precision–Recall curves (Figures 5.17 and 5.18) provide the definitive quantitative comparison of loop-closure detection. Both configurations achieve near-perfect precision (**AlexNet: 0.996**, **ShuffleNetV2: 0.997**), indicating that false positives are rare—a critical requirement for preventing map corruption. At the operating point used to compute summary statistics, the ShuffleNetV2 configuration achieves a higher F1-score (**F1=0.010**) than AlexNet-conv3 (**F1=0.008**), reflecting improved balance between precision and recall under this challenging, aliasing-prone scenario.

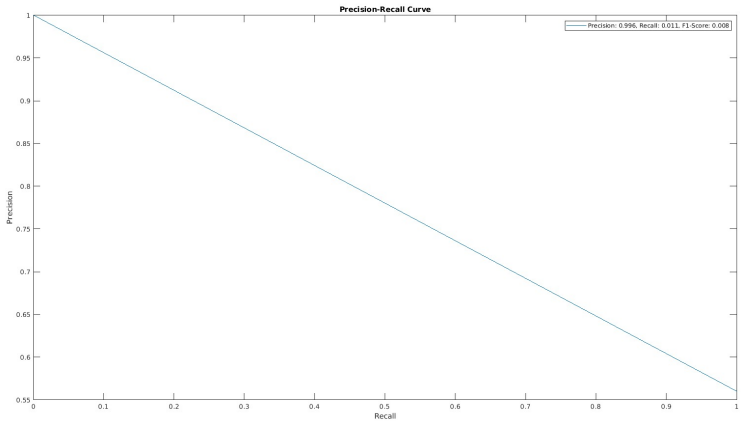


Figure 5.17: Precision–Recall curve for NeoSLAM using AlexNet-conv3. Source: The author.

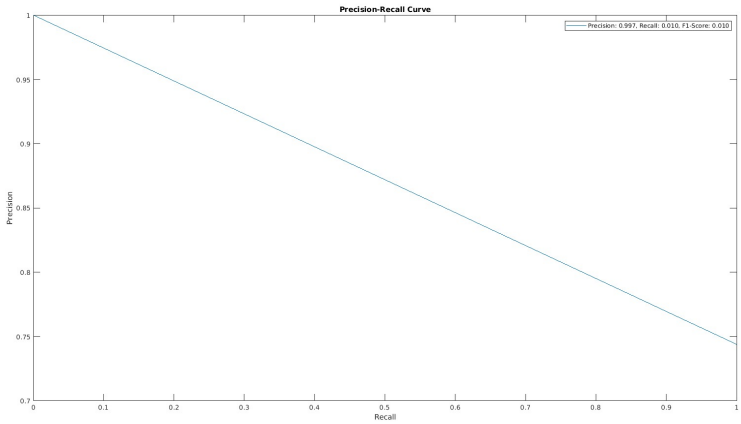


Figure 5.18: Precision–Recall curve for NeoSLAM using ShuffleNetV2 x1.0. The higher F1-score indicates a better balance of reliability and sensitivity relative to Figure 5.17. Source: The author.

5.4.4 PoseidonSLAM: VPR Guard ON vs OFF Ablation

This subsection incorporates the **PoseidonSLAM** results. The objective is **not** to claim full localization accuracy; instead, we (i) validate **traceability** between each topological

event and its corresponding perceptual evidence, and (ii) quantify how the **VPR Guard** mechanism modulates **topological sampling density** and **3D map stability proxies**. The comparison uses the same descriptor source (`tm`) in both regimes and an event–image audit tolerance (`slop_sec`) of 0.750 s.

Audit of Event–Evidence Correspondence

PoseidonSLAM emits an experience creation event (`/experience_event`) alongside an associated perceptual evidence message (`/experience_image`). We audited the one-to-one correspondence between these topics to ensure that every topological update can be traced to a unique sensory snapshot. The audit achieves perfect association in both ablations: precision = recall = F1 = 1.000, with no missing events or missing images. This result is essential for experimental rigor because it guarantees that any differences in map structure are attributable to algorithmic choices (e.g., VPR Guard gating) rather than logging or synchronization artifacts.

Loop-Closure Evidence by GT Distance

To classify each accepted loop closure, we compute the GT distance $d_{\text{gt}} = \|\mathbf{g}_{\text{current}} - \mathbf{g}_{\text{matched}}\|_2$ between the GT marker positions associated with the current and matched experience IDs. We adopt $d_{\text{thr}} = 5.0$ m: acceptances with $d_{\text{gt}} \leq d_{\text{thr}}$ are labeled TP; otherwise FP. Tables 5.2 and 5.3 summarize the accepted events. VPR Guard **ON** yields 2 TPs; VPR Guard **OFF** yields only FPs.

event_seq	current_exp	matched_exp	d_{gt} (m)	current_gt (x,y,z)	matched_gt (x,y,z)	Label
520	520	136	2.213	(-38.948, 52.478, 16.738)	(-39.709, 50.781, 15.539)	TP
555	555	91	1.765	(-26.074, 36.799, 15.347)	(-24.716, 37.865, 14.983)	TP

Table 5.2: Accepted loop closures for VPR Guard ON. Both events satisfy $d_{\text{gt}} \leq 5.0$ m.

Table 5.3: Accepted loop closures for VPR Guard OFF. All events exceed $d_{\text{gt}} = 5.0$ m and are labeled FP.

event_seq	current_exp	matched_exp	d_{gt} (m)	current_gt (x,y,z)	matched_gt (x,y,z)	Label
805	805	493	7.625	(-83.555, 43.875, 8.100)	(-76.321, 45.342, 10.012)	FP
884	884	370	15.491	(-68.006, 48.150, 11.980)	(-52.950, 48.319, 15.623)	FP
923	923	363	10.325	(-61.102, 51.251, 13.512)	(-51.491, 48.218, 15.756)	FP
945	945	307	16.927	(-56.619, 53.361, 14.202)	(-39.960, 50.707, 15.597)	FP
985	985	212	28.380	(-48.624, 56.200, 15.599)	(-24.972, 40.531, 14.897)	FP
1155	1155	283	36.777	(-60.248, 76.265, 16.494)	(-34.615, 49.933, 15.007)	FP
1181	1181	370	31.573	(-59.362, 79.222, 14.785)	(-52.950, 48.319, 15.623)	FP
1209	1209	332	33.077	(-56.008, 80.670, 15.269)	(-45.098, 49.459, 16.227)	FP
1240	1240	388	33.509	(-49.382, 80.784, 16.730)	(-56.756, 48.133, 15.181)	FP
1268	1268	388	35.594	(-43.758, 81.168, 17.753)	(-56.756, 48.133, 15.181)	FP
1288	1288	332	31.139	(-38.962, 79.922, 18.224)	(-45.098, 49.459, 16.227)	FP

Continues on next page

Table 5.3 (continued)

event_seq	current_exp	matched_exp	d_{gt} (m)	current_gt (x,y,z)	matched_gt (x,y,z)	Label
1335	1335	1025	18.263	(-32.769, 75.134, 19.016)	(-41.644, 59.272, 17.234)	FP
2174	2174	1920	8.618	(-49.512, 5.101, 8.883)	(-54.561, -1.507, 6.620)	FP
2213	2213	1781	14.409	(-40.751, 6.282, 10.536)	(-26.853, 9.243, 12.917)	FP

Topological Sampling Rate and Graph Size

Table 5.4 summarizes the core ON/OFF metrics extracted from the comparison outputs in `compare_plots_223758_vs_231839/paper_outputs`. Disabling the guard increases event density from 0.404 Hz (ON) to 1.239 Hz (OFF), producing a substantially larger graph (786 vs 2,413 nodes and 786 vs 2,414 edges). The mean degree remains ≈2.0 in both cases, consistent with a predominantly chain-like incremental topology in which loop-closure constraints primarily *warp* the map geometry rather than increasing average connectivity.

Table 5.4: PoseidonSLAM ablation study (VPR Guard ON vs OFF): core audit metrics, sampling density, and best-effort stability proxies (association metrics; not a substitute for full localization evaluation). Values extracted from the comparison outputs.

Metric	VPR Guard ON	VPR Guard OFF
Event density (events/s)	0.403721	1.238656
Nodes/Edges (final)	786 / 786	2413 / 2414
Mean degree (final)	2.000000	2.000829
Audit F1 (event↔image)	1.000000	1.000000
Δt_{real} mean / max (s)	0.020634 / 0.236000	0.020082 / 0.256000
VPR window ok_ratio	0.258289	0.248225
Span norm ratio MAP/ODOM	0.981359	1.018186
Volume ratio MAP/ODOM	0.947524	0.921947
End-disp ratio MAP/ODOM	5.041357	4.534118
Span norm ratio MAP/GT	0.922785	0.950239
Volume ratio MAP/GT	0.842433	0.809499
End-disp ratio MAP/GT	0.864895	0.752254

3D Trajectory and Topological Geometry (Qualitative)

Figures 5.19 and 5.20 provide qualitative comparisons against ground truth and dead reckoning. For transparency and reproducibility of the qualitative assessment, companion videos of the full PoseidonSLAM runs under **VPR Guard ON** and **VPR Guard OFF** are provided online[3]. The 3D overlay (Figure 5.19) shows that both PoseidonSLAM configurations alter the dead-reckoning trajectory, but the **VPR Guard ON** mode yields a trajectory that remains visually closer to the ground-truth structure in key regions, while

[3] https://youtu.be/dX24Jc5Gj-8 (VPR Guard ON) and https://youtu.be/efMvf-D91qo (VPR Guard OFF)

VPR Guard OFF exhibits larger deviations in segments where perceptual aliasing and dense event emission may amplify inconsistent corrections. The planar projections (Figure 5.20) disentangle this effect across XY, XZ, and YZ planes, revealing where corrections occur predominantly in horizontal alignment versus vertical structure.

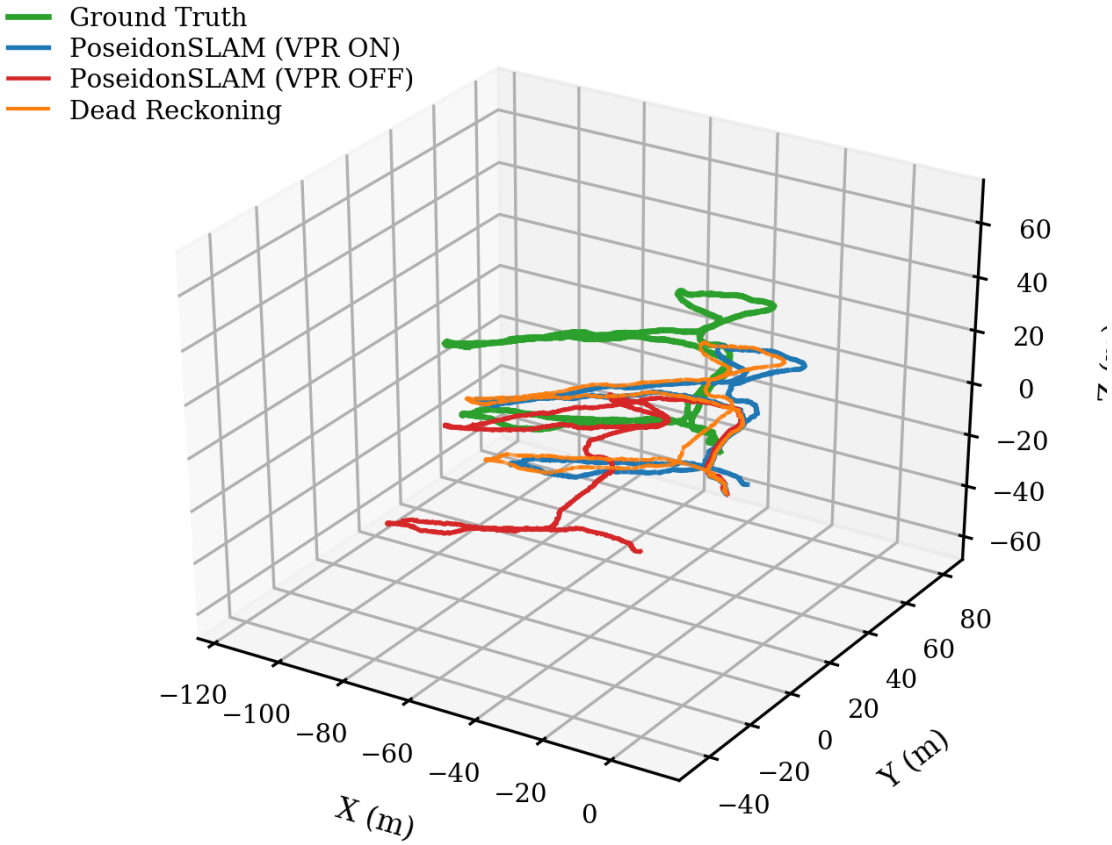


Figure 5.19: Three-dimensional trajectory comparison between **ground truth** (green), **dead reckoning** (orange), and **PoseidonSLAM** with **VPR Guard ON** (blue) and **OFF** (red). The overlay highlights how VPR Guard changes the sequence of topological corrections applied to dead reckoning: VPR Guard ON produces a less densely updated trajectory with visually closer adherence to the ground-truth structure in key segments, whereas VPR Guard OFF shows larger deviations consistent with denser event emission.

To isolate the final topological snapshot under each regime, Figure 5.21 overlays the final PoseidonSLAM map, dead reckoning, and ground truth separately for VPR Guard ON and VPR Guard OFF. The visual density difference is consistent with event-count statistics: VPR Guard OFF produces a denser trace with more frequent experience insertions, whereas VPR Guard ON yields a sparser topological sampling while preserving a non-collapsed 3D structure.

Map Deformation Proxies and Stability Trade-offs

To quantify structural effects beyond visual inspection, we report best-effort deformation proxies (Table 5.4). Ratios such as span norms and volumes compare the global extent

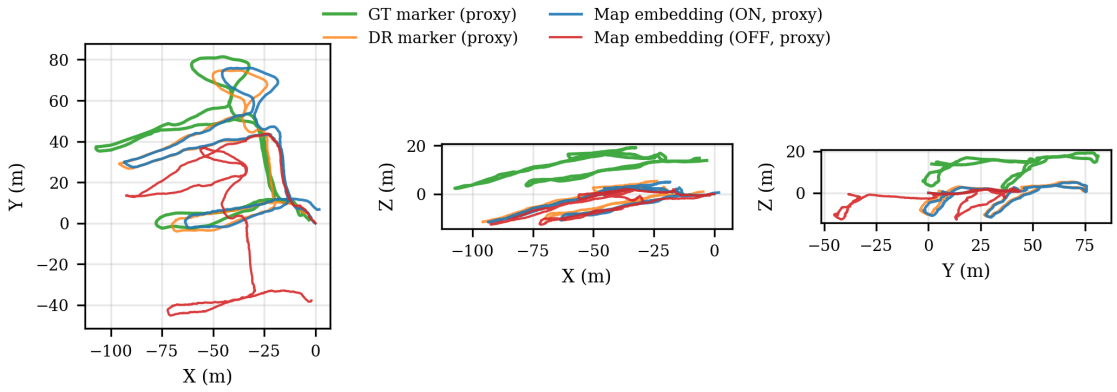
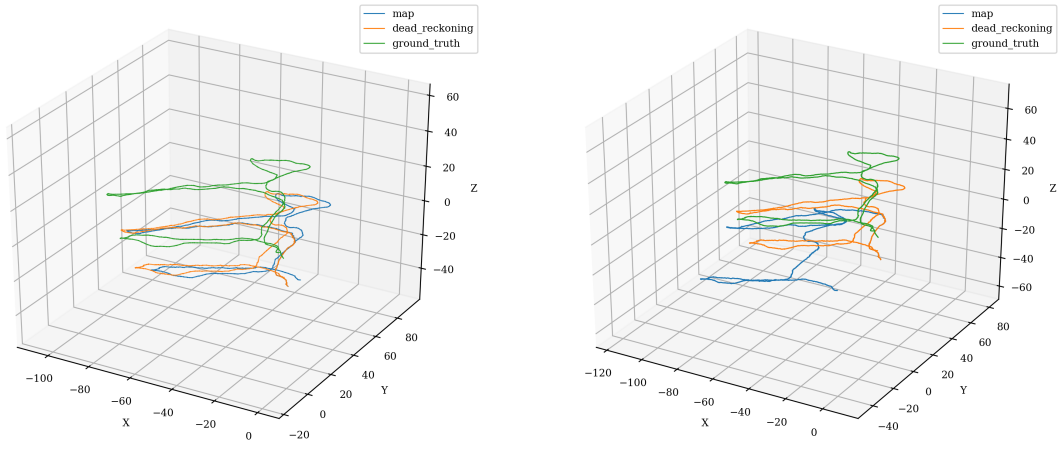


Figure 5.20: Planar projections of map embeddings (proxy; not time-aligned). The XY projection emphasizes horizontal alignment and loop geometry; XZ and YZ projections reveal how vertical structure differs between dead reckoning and PoseidonSLAM corrections. These views support qualitative assessment of where each ablation concentrates corrections and where drift remains.



(a) VPR Guard ON

(b) VPR Guard OFF

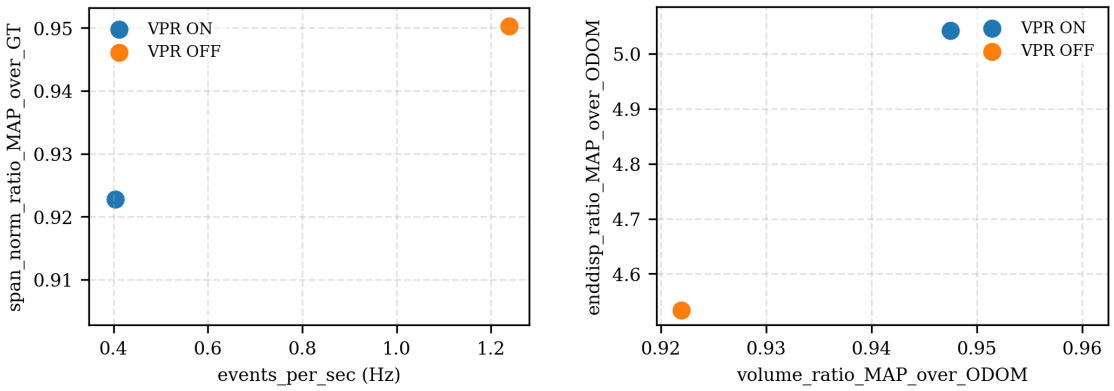
Figure 5.21: Final 3D topology overlays for PoseidonSLAM under VPR Guard ON and OFF. Each subplot overlays the final **map** estimate, **dead reckoning**, and **ground truth**. The OFF mode exhibits a visually denser trace consistent with higher event density and larger graph size, while the ON mode yields a sparser topological sampling without collapse indicators.

of the final map against reference trajectories (odometry or ground truth). Both regimes remain close to reference scale (span norm ratios near 1) and do not exhibit collapse (span-drop ratios at or near 1). However, **VPR Guard ON** yields a higher volume ratio MAP/ODOM (0.948 vs 0.922), i.e., a less compact spatial envelope relative to odometry, and a higher end-displacement ratio MAP/ODOM (5.041 vs 4.534), indicating stronger endpoint deviation relative to odometry.

This trade-off is summarized in the stability scatter (Figure 5.22b), which places each mode in the plane of (volume ratio, end-displacement ratio). The plot highlights that VPR Guard OFF preserves a larger volumetric spread (closer to odometry) while exhibiting a lower endpoint discrepancy, whereas VPR Guard ON compresses the volume more strongly while increasing endpoint deviation. Importantly, these are *structural proxies* of topological map deformation and should not be conflated with standard localization error metrics.

Event Density vs. GT Alignment and Effective Rate

Figure 5.22a relates event density to a GT-based span-norm ratio. The VPR Guard OFF mode achieves higher span-norm ratio MAP/GT (0.950 vs 0.923) but at substantially higher event rate. Thus, within this proxy, the ablation suggests a *sampling* trade-off: denser topological updates may marginally improve global alignment to GT while increasing computational and representation cost (graph size).



(a) Event density vs. GT alignment proxy (b) Volume vs. endpoint deformation proxy

Figure 5.22: Proxy scatters summarizing the ON/OFF trade-off. Left: higher event density (OFF) corresponds to a slightly higher span-norm ratio MAP/GT but at substantially increased sampling rate. Right: VPR Guard ON yields a more compact map volume relative to odometry, while VPR Guard OFF preserves a larger volume ratio and exhibits a lower end-displacement ratio. These are structural proxies and do not replace standard localization metrics.

We also report the implied effective rate derived from the mean bag-time spacing statistic ($\Delta t_{\text{real,mean}}$) between matched event–image pairs. Inverting this quantity yields an *effective message-rate proxy* of approximately 49.8 Hz (OFF) versus 48.5 Hz (ON), plotted in Figure 5.23. This quantity should be interpreted as a rate proxy tied to the temporal spacing of the processed stream, not as a hardware-benchmark of compute time.

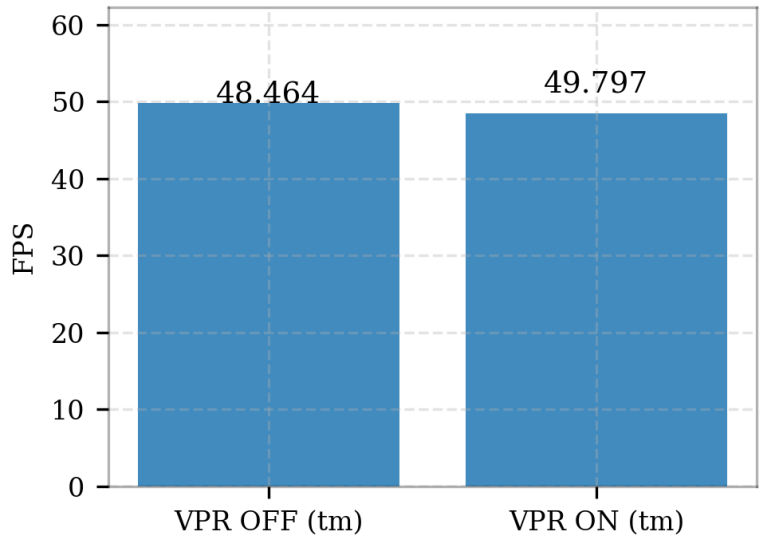


Figure 5.23: Implied effective rate proxy computed as $1/\Delta t_{\text{real,mean}}$, where $\Delta t_{\text{real,mean}}$ is the mean bag-time spacing statistic reported by the event–image audit. OFF yields a higher implied rate than ON, consistent with denser event emission. This plot should be interpreted as a stream-rate proxy rather than a hardware runtime benchmark.

Proxy Spatial Deviation (Map Embedding vs. GT Marker)

Although the primary focus of PoseidonSLAM is topological stability rather than full metric localization, we include **proxy spatial deviation** plots computed from map/marker embeddings (*not time-aligned*) as qualitative diagnostics. Figure 5.24a shows the temporal evolution of this proxy: **VPR Guard OFF** reaches larger deviations and exhibits spikes, while **VPR Guard ON** remains lower and smoother. Figure 5.24b summarizes the proxy distribution.

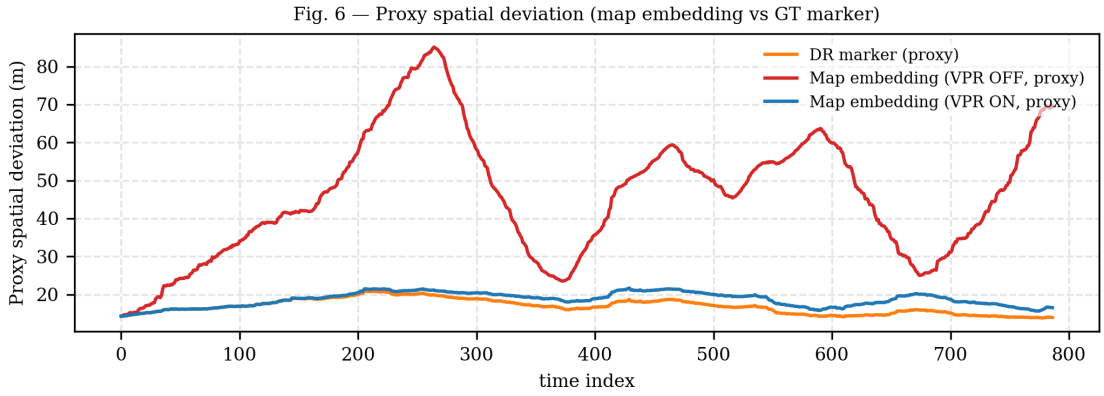
Figure 5.25 projects the same proxy deviation onto the XY plane, indicating regions where the embedding deviation concentrates.

5.5 Discussion of Experimental Findings

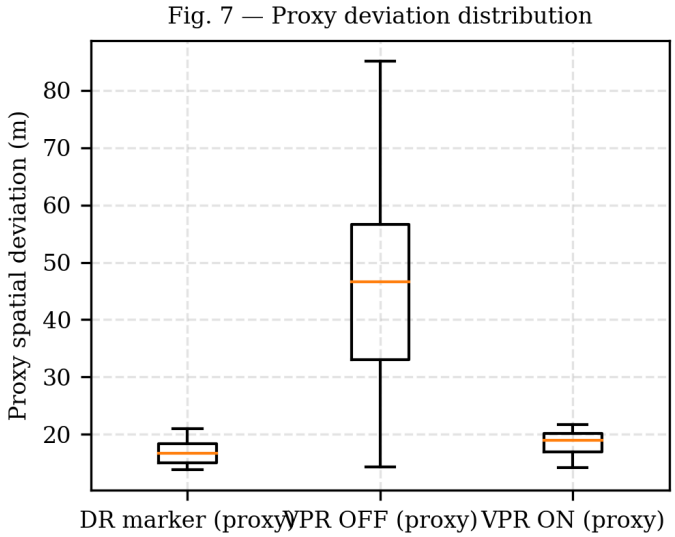
The experiments support three main conclusions.

(1) Baseline fragility in underwater perception. ORB-SLAM3 fails catastrophically due to the lack of stable, repeatable features in turbid, low-texture imagery. OpenRatSLAM, despite principled tuning, collapses under perceptual aliasing, demonstrating that SAD-based template matching is brittle in this domain.

(2) Neocortical-inspired perception enables robust topological mapping. NeoSLAM successfully constructs a coherent Experience Map over the full trajectory. Similarity matrices, View Cell statistics, and trajectory plots indicate that the CNN front-end critically affects perceptual discriminability. ShuffleNetV2 x1.0 produces fewer landmarks yet more loop closures than AlexNet-conv3, evidencing improved efficiency and robustness.



(a) Proxy spatial deviation over time index (map embedding vs. GT marker; not time-aligned)



(b) Proxy spatial deviation distribution (boxplot)

Figure 5.24: Proxy spatial deviation computed from map/marker embeddings (not time-aligned). These plots are qualitative and do not represent true ATE.

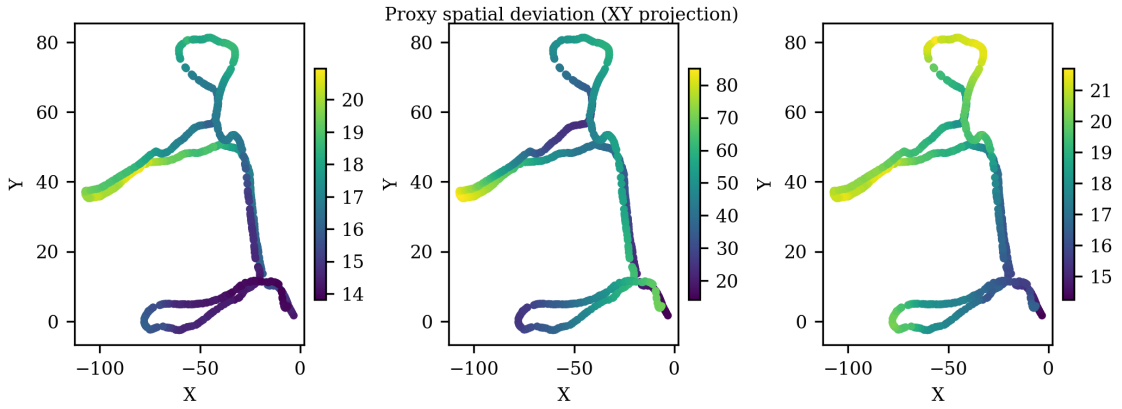


Figure 5.25: Spatial distribution of proxy deviation on the XY plane for (left) dead reckoning marker, (middle) PoseidonSLAM VPR Guard OFF, and (right) PoseidonSLAM VPR Guard ON.

(3) PoseidonSLAM VPR Guard regulates sampling and reveals stability trade-offs. The PoseidonSLAM ablation demonstrates that enabling the VPR Guard reduces event density by approximately 63% and produces a much smaller graph while preserving perfect event–evidence traceability (audit F1=1.0). Stability proxies show that both regimes avoid collapse, but they occupy different regions in the (volume, endpoint deformation) plane. This suggests that the guard can be used as a *topological sampling regulator*, allowing practitioners to trade graph compactness and correction aggressiveness against denser sampling and slightly different global alignment behavior.

5.5.1 Computational Efficiency as a Driver for Perceptual Robustness

ShuffleNetV2 was designed around practical efficiency guidelines that reduce memory access cost and fragmentation [12]. Our results suggest that such efficiency can correlate with more discriminative features in degraded imagery, improving loop-closure yield and stabilizing the downstream topological map. This reinforces the design principle that perceptual robustness can emerge from architectural intelligence rather than brute-force capacity.

5.5.2 Implications for Bio-Inspired SLAM System Design

These findings emphasize co-design of the perceptual front-end and spatial memory backend. The hippocampal-inspired Pose Cells and Experience Map provide a robust substrate for spatial memory [8], but overall performance remains strongly conditioned on perceptual input quality. PoseidonSLAM extends this pipeline by explicitly auditing event–evidence correspondence and introducing a VPR Guard that controls when new experiences are committed. This combination is particularly relevant for long-term underwater autonomy, where perceptual aliasing and intermittent sensing can otherwise corrupt topological memory.

Chapter 6

Conclusion

This final chapter synthesizes the contributions of this dissertation and positions them within the broader context of underwater robotic autonomy. We consolidate the key empirical findings, articulate the scope and limitations of the validation, and outline a research agenda that builds directly upon the methods, artifacts, and insights established herein.

6.1 Concluding Remarks

This dissertation addressed the critical challenge of long-term SLAM in perceptually degraded, GPS-denied underwater environments, where navigation must remain reliable despite turbidity, low texture, illumination variability, and strong perceptual aliasing. Through a combination of conceptual analysis and empirical evaluation, we demonstrated that the two dominant families of SLAM approaches exhibit fundamental failure modes under such conditions. On the one hand, state-of-the-art geometric methods (represented by ORB-SLAM3) are vulnerable to catastrophic breakdown when their discrete feature detection and matching assumptions are violated by visually ambiguous imagery [4]. On the other hand, canonical bio-inspired approaches (exemplified by RatSLAM) may also collapse, since low-resolution template matching remains brittle in the presence of extreme aliasing, leading to topologically incoherent maps [8].

The central hypothesis investigated in this dissertation is that a neocortical-inspired sequence-processing mechanism can mitigate these dual limitations by providing a representational substrate that is intrinsically robust to ambiguous instantaneous observations. The experimental results support this hypothesis: using a challenging transformed benchmark derived from a real underwater cave experiment, we showed that the proposed NeoSLAM framework sustains coherent topological mapping in conditions where both ORB-SLAM3 and RatSLAM fail. NeoSLAM leverages Hierarchical Temporal Memory principles and Sparse Distributed Representations to explicitly encode temporal context and sequence consistency, thereby reducing susceptibility to perceptual aliasing and short-lived appearance changes [70].

Beyond demonstrating robustness at the architectural level, this dissertation pro-

vided quantitative evidence that the choice of visual front-end critically affects long-term place recognition and loop-closure yield. In particular, the comparative analysis between AlexNet-conv3 and ShuffleNetV2 x1.0 indicated that the lightweight ShuffleNetV2 front-end achieved a superior balance of precision and recall (as reflected by the reported F1-score improvement), resulting in more consistent loop-closure opportunities and a more stable final map. This observation supports an important design implication: in degraded domains, computational efficiency in the perceptual front-end is not merely a practical constraint, but can correlate with improved discriminability and robustness when coupled to a temporally grounded back-end [12].

Crucially, the dissertation advances from "visual-only" robustness toward *Visual-Inertial-Acoustic Underwater SLAM* by incorporating PoseidonSLAM as the applied, system-level instantiation of the POSEIDON NAV principles. The PoseidonSLAM results demonstrate traceable, audit-consistent coupling between topological events and perceptual evidence (F1=1.0) and quantify how experience gating modulates graph growth and event density (0.404 Hz and 786/786 nodes/edges with *VPR Guard ON* versus 1.239 Hz and 2413/2414 nodes/edges with *OFF*). Importantly, loop-closure evidence classified by GT distance shows *VPR Guard ON* yielding 2 TP and 0 FP ($d_{\text{gt}} \leq 5.0$ m), while *OFF* yields 14 FP ($d_{\text{gt}} > 5.0$ m). These outcomes matter for VIA SLAM because they show that controlled experience insertion can regulate event density and graph size while preventing spurious visual matches from propagating as destabilizing constraints in the fused estimate.

Finally, this dissertation introduced the *POSEIDON NAV Project* as a comprehensive architectural direction for robust underwater autonomy. POSEIDON NAV formalizes the synergistic combination of (i) neocortical-inspired temporal verification, (ii) efficient deep visual descriptors, and (iii) multimodal priors (inertial and acoustic) to realize robust long-term mapping and navigation in 3D. Within this framing, NeoSLAM provides the core temporal reasoning substrate, while PoseidonSLAM operationalizes these principles in a system context aligned with Visual-Inertial-Acoustic Underwater SLAM.

6.2 Summary of Contributions

The contributions of this dissertation can be summarized as follows:

1. **A transformed underwater benchmark for long-term robustness evaluation.** We curated, merged, and re-structured a real underwater cave dataset into a transformed multi-loop benchmark that simulates repeated traversals under controlled viewpoint perturbations, enabling systematic evaluation of long-term VPR and loop-closure robustness.

2. **A neocortical-inspired Visual SLAM framework (NeoSLAM) validated under underwater perceptual degradation.** We implemented and evaluated NeoSLAM using HTM/SDR principles, demonstrating coherent topological mapping

where a geometric baseline (ORB-SLAM3) and a canonical bio-inspired baseline (RatSLAM) fail.

3. **A controlled comparison of CNN front-ends for temporally grounded VPR.** We compared AlexNet-conv3 and ShuffleNetV2 x1.0 as perceptual encoders, quantitatively linking front-end choice to loop-closure yield and downstream map stability.

4. **POSEIDON NAV as an architectural pathway toward 3D multimodal autonomy.** We introduced POSEIDON NAV as a modular system direction for extending neocortical-inspired VPR and topological mapping into 3D and into a visual-inertial-acoustic navigation stack.

5. **PoseidonSLAM as a system-level validation aligned with VIA Underwater SLAM.** We incorporated an event–evidence audit (traceability), a guard-based experience insertion policy (VPR Guard), and stability-focused metrics/visualizations, reporting ablation results that characterize how experience gating modulates event density, graph growth, and loop-closure integrity: *VPR Guard ON* yields 2 TP and 0 FP, whereas *OFF* yields 14 FP (all with $d_{\text{gt}} > 5.0$ m).

6.3 Limitations

Academic rigor requires that the scope of the conclusions be bounded by the limitations of the evaluation. The principal limitations of this dissertation are:

- **Dataset scope and generalization.** Although the benchmark originates from a real underwater experiment, the transformed long-term scenario relies on programmatic loop replication and controlled viewpoint perturbations. The conclusions therefore pertain to a well-defined class of appearance variations and should be further tested across diverse underwater environments, sensors, and mission profiles.

- **Software-in-the-loop experimental validation** The evaluation was conducted through a software-in-the-loop protocol with prerecorded ROS bag streams and transformed benchmark artifacts. It does not fully model real vehicle dynamics, complex sensor noise characteristics (e.g., IMU/DVL biases and scale drift), communication delays, or environmental factors such as suspended particles, severe lighting changes, occlusions, scale variations, and motion-blur effects that occur in field deployments.

- **Topological emphasis and bounded metric claims.** The dissertation's primary claim concerns perceptual robustness, sequence verification, and coherent topological mapping. The reported stability proxies (span/volume/end-displacement ratios) and proxy spatial-deviation plots (map embedding vs GT, not time-aligned) do not constitute a complete metric SLAM accuracy assessment.

- **Visual-Inertial-Acoustic coupling is supported, but not exhaustively benchmarked.** PoseidonSLAM demonstrates solid results consistent with VIA operation and provides evidence that visual constraint injection can be audited and regulated. However, the dissertation does not yet deliver a full, tightly controlled benchmarking campaign that isolates the individual contribution of inertial and acoustic modalities (e.g., ablations for IMU-only, acoustic-only, and full VIA), nor does it provide complete calibration/latency sensitivity analyses under varied field conditions.
- **Runtime characterization is indicative rather than hardware-grounded.** The reported FPS proxy derived from bag-time spacing reflects stream timing characteristics and relative throughput under the logging conditions; it should not be interpreted as a controlled compute-time benchmark across heterogeneous onboard platforms.

6.4 Future Work

The findings and limitations of this dissertation motivate a concrete research agenda aimed at transitioning the presented framework from a robust proof-of-concept into a field-ready solution for visual-inertial-acoustic underwater autonomy.

(1) Large-scale validation across datasets and conditions. A first priority is replicating the evaluation on multiple underwater datasets (different caves, reefs, and infrastructure sites), including multi-session data when available. This should include statistically grounded reporting (multiple runs, confidence intervals, and sensitivity analyses) to quantify robustness and failure boundaries.

(2) Full Visual-Inertial-Acoustic benchmarking and modality ablations. To substantiate the VIA claim rigorously, future work should execute a structured evaluation protocol that includes: (i) visual-only, (ii) visual–inertial, (iii) visual–acoustic, and (iv) full visual-inertial-acoustic configurations, under matched datasets and identical loop-closure policies. This should be complemented by calibration and synchronization studies (time offsets, clock drift, latency) to quantify how fusion quality depends on real sensor timing.

(3) Systematic evaluation and design of VPR Guard policies. Beyond ON/OFF ablation, future work should investigate adaptive experience gating policies that respond to uncertainty, novelty, and sequence-consistency scores. The evaluation should report how gating affects false positives, graph growth, map deformation, and downstream state-estimation stability in the fused VIA pipeline.

(4) Hybrid perceptual architectures for higher recall without sacrificing precision. To address persistent low recall in severely degraded underwater imagery, future research should explore hybrid VPR pipelines: high-recall candidate generation using stronger global descriptors or foundation models, followed by NeoSLAM-style temporal verification for precision-preserving acceptance. Lightweight semantics (repeatable seabed patterns, man-made structures, persistent landmarks) should be integrated as auxiliary cues for disambiguation.

(5) Lifelong mapping and multi-map management in underwater missions. A long-term extension is to support lifelong SLAM, enabling map updates across missions and environmental changes. Inspired by multi-map strategies such as Atlas-style mapping [4], POSEIDON NAV should be extended to manage multiple map hypotheses, merge compatible topological submaps, and retain long-term stability through selective consolidation and forgetting mechanisms.

(6) Field deployment and closed-loop autonomy. The decisive validation step is deployment on a physical AUV with real dynamics and noise sources. This includes benchmarking end-to-end navigation performance in closed-loop operation (mission execution, revisitation, and return-to-start behaviors), evaluating resilience to disturbances and dropouts, and quantifying practical autonomy metrics such as mission completion rate, map reuse across days, and operational envelope in varying turbidity and lighting.

Additional future work should include: (i) an explicit OR vs. AND multimodal fusion ablation; (ii) richer appearance perturbations in the transformed benchmark, including illumination, particles, occlusions, and scale changes; and (iii) image-level qualitative inspection panels to document true positives, false positives, false negatives, and ambiguous revisitation cases.

In summary, this dissertation provides evidence that neocortical-inspired temporal processing, efficient visual descriptors, and controlled memory updates form a compelling foundation for robust underwater SLAM. With the addition of PoseidonSLAM results, the dissertation also supports a concrete and technically grounded pathway toward Visual-Inertial-Acoustic Underwater SLAM: visual place recognition becomes *auditable and regulatable* under severe aliasing, enabling safer integration of loop-closure constraints into multimodal fused estimates. This foundation positions POSEIDON NAV as a viable research program toward persistent, long-term underwater autonomy.

Bibliography

[1] WU, H., CHEN, Y., YANG, Q., et al. "A review of underwater robot localization in confined spaces", *Journal of Marine Science and Engineering*, v. 12, n. 3, pp. 428, 2024.

[2] QIN, J., LI, M., LI, D., et al. "A survey on visual navigation and positioning for autonomous UUVs", *Remote Sensing*, v. 14, n. 15, pp. 3794, 2022.

[3] ZHANG, S., ZHAO, S., AN, D., et al. "Visual SLAM for underwater vehicles: A survey", *Computer Science Review*, v. 46, pp. 100510, 2022.

[4] CAMPOS, C., ELVIRA, R., RODRÍGUEZ, J. J. G., et al. "Orb-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM", *IEEE transactions on robotics*, v. 37, n. 6, pp. 1874–1890, 2021.

[5] MALLIOS, A., RIDAO, P., RIBAS, D., et al. "Toward autonomous exploration in confined underwater environments", *Journal of Field Robotics*, v. 33, n. 7, pp. 994–1012, 2016.

[6] MALLIOS, A., VIDAL, E., CAMPOS, R., et al. "Underwater caves sonar data set", *The International Journal of Robotics Research*, v. 36, n. 12, pp. 1247–1251, 2017.

[7] MILFORD, M. J., WYETH, G. F., PRASSER, D. "RatSLAM: a hippocampal model for simultaneous localization and mapping". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, v. 1, pp. 403–408. IEEE, 2004.

[8] BALL, D., HEATH, S., WILES, J., et al. "OpenRatSLAM: an open source brain-based SLAM system", *Autonomous Robots*, v. 34, pp. 149–176, 2013.

[9] PIZZINO, C. A. P., COSTA, R. R., MITCHELL, D., et al. "NeoSLAM: Long-term SLAM using computational models of the brain", *Sensors*, v. 24, n. 4, pp. 1143, 2024.

[10] PIZZINO, C. A. P. "NeoSLAM: Long-Term SLAM Using Computational Models of the Brain", PhD Thesis, 2024, 2024.

[11] QUIGLEY, M., CONLEY, K., GERKEY, B., et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*, v. 3, p. 5. Kobe, Japan, 2009.

[12] MA, N., ZHANG, X., ZHENG, H.-T., et al. "Shufflenet v2: Practical guidelines for efficient CNN architecture design". In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

[13] ZHANG, X., ZHOU, X., LIN, M., et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.

[14] KRIZHEVSKY, A., SUTSKEVER, I., HINTON, G. E. "Imagenet classification with deep convolutional neural networks", *Advances in neural information processing systems*, v. 25, 2012.

[15] LOWE, D. G. "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, v. 60, n. 2, pp. 91–110, 2004.

[16] BAY, H., ESS, A., TUYTELAARS, T., et al. "Speeded-up robust features (SURF)", *Computer Vision and Image Understanding*, v. 110, n. 3, pp. 346–359, 2008.

[17] RUBLEE, E., RABAUD, V., KONOLIGE, K., et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International conference on computer vision*, pp. 2564–2571. Ieee, 2011.

[18] IQBAL, A., GANS, N. R. "Data association and localization of classified objects in visual SLAM", *Journal of Intelligent & Robotic Systems*, v. 100, n. 1, pp. 113–130, 2020.

[19] GALDRAN, A., PARDO, D., PICÓN, A., et al. "Automatic red-channel underwater image restoration", *Journal of Visual Communication and Image Representation*, v. 26, pp. 132–145, 2015.

[20] SCHETTINI, R., CORCHS, S. "Underwater image processing: state of the art of restoration and image enhancement methods", *EURASIP journal on advances in signal processing*, v. 2010, pp. 1–14, 2010.

[21] HUANG, H., LIU, Z., ZHANG, L., et al. "Via-SLAM: An underwater visual-inertial-acoustic SLAM with integrated DVL", *IEEE Transactions on Instrumentation and Measurement*, 2025.

[22] EUSTICE, R., PIZARRO, O., SINGH, H. "Visually augmented navigation for autonomous underwater vehicles", *IEEE Journal of Oceanic Engineering*, v. 33, pp. 103–122, 2008.

[23] LI, S., NI, P. "Square-root unscented Kalman filter based simultaneous localization and mapping". In: *The 2010 IEEE International Conference on Information and Automation*, pp. 2384–2388. IEEE, 2010.

[24] MAKI, T., KONDO, H., URA, T., et al. "Photo mosaicing of Tagiri shallow vent area by the AUV Tri-Dog 1 using a SLAM based navigation scheme". In: *OCEANS 2006*, pp. 1–6. IEEE, 2006.

[25] DURRANT-WHYTE, H., BAILEY, T. "Simultaneous localization and mapping: part I", *IEEE robotics & automation magazine*, v. 13, n. 2, pp. 99–110, 2006.

[26] DUBBELMAN, G., BROWNING, B. "COP-SLAM: Closed-form online pose-chain optimization for visual SLAM", *IEEE Transactions on Robotics*, v. 31, n. 5, pp. 1194–1213, 2015.

[27] MUR-ARTAL, R., MONTIEL, J. M., TARDOS, J. D. "ORB-SLAM: a versatile and accurate monocular SLAM system", *IEEE Transactions on Robotics*, v. 31, n. 5, pp. 1147–1163, 2015.

[28] KIM, A., EUSTICE, R. M. "Real-time visual SLAM for autonomous underwater hull inspection using visual saliency", *IEEE Transactions on Robotics*, v. 29, pp. 719–733, 2013.

[29] VARGAS, E., SCONA, R., WILLNERS, J., et al. "Robust underwater visual SLAM fusing acoustic sensing". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2140–2146. IEEE, 2021.

[30] GALVEZ-LOPEZ, D., TARDOS, J. D. "Bags of binary words for fast place recognition in image sequences", *IEEE Transactions on Robotics*, v. 28, n. 5, pp. 1188–1197, 2012.

[31] NEGRE CARRASCO, P. L., BONIN-FONT, F., OLIVER-CODINA, G. "Global image signature for visual loop-closure detection", *Autonomous Robots*, v. 40, pp. 1403–1417, 2016.

[32] BURGUERA, A., BONIN-FONT, F. "An unsupervised neural network for loop detection in underwater visual SLAM", *Journal of Intelligent & Robotic Systems*, v. 100, n. 3, pp. 1157–1177, 2020.

[33] BONIN-FONT, F., BURGUERA BURGUERA, A. "NetHALOC: A learned global image descriptor for loop closing in underwater visual SLAM", *Expert Systems*, v. 38, n. 2, pp. e12635, 2021.

[34] SONG, J., LI, W., ZHU, X. "Acoustic-VINS: Tightly coupled acoustic-visual-inertial navigation system for autonomous underwater vehicles", *IEEE Robotics and Automation Letters*, v. 9, n. 2, pp. 1620–1627, 2023.

[35] RIDAO, P., RIBAS, D., TENA, I., et al. "Probabilistic underwater 3D SLAM with a mechanically scanned imaging sonar", *Sensors*, v. 15, n. 2, pp. 30707–30737, 2015.

[36] VALLVE, J., RIBAS, D., RIDAO, P. "Towards multi-session feature-based SLAM for AUVs". In: *IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*, pp. 1–7. IEEE, 2018.

[37] ZHANG, J., HAN, F., HAN, D., et al. "Integration of Sonar and Visual–Inertial Systems for SLAM in Underwater Environments", *IEEE Sensors Journal*, v. 24, n. 10, pp. 16792–16803, 2024. doi: 10.1109/JSEN.2024.3384301.

[38] RAHMAN, S., LI, A. Q., REKLEITIS, I. "Svin2: An underwater SLAM systemusing sonar, visual, inertial, and depth sensor". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1861–1868. IEEE, 2019.

[39] JOSHI, B., REKLEITIS, I. "Enhancing Visual Inertial SLAM with Magnetic Measurements", *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, p. 10012, 2024. doi: 10.1109/ICRA57147.2024.10611341.

[40] MCDONALD, J. B. *Multi-session Visual Simultaneous Localisation and Mapping*. Tese de Doutorado, National University of Ireland, Maynooth (Ireland), 2013.

[41] JANG, H., YOON, S., KIM, A. "Multi-session underwater pose-graph SLAM using inter-session opti-acoustic two-view factor". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11668–11674. IEEE, 2021.

[42] TEIXEIRA, B., SILVA, H., MATOS, A., et al. "Deep Learning for Underwater Visual Odometry Estimation", *IEEE Access*, v. 8, pp. 44687–44697, 2020. doi: 10.1109/ACCESS.2020.2978406.

[43] MENG, H., LU, H. "A Survey of Deep Learning Technology in Visual SLAM", *Proceedings of the International Wireless Communications and Mobile Computing Conference*, p. 37, 2024. doi: 10.1109/IWCMC61514.2024.10592584.

[44] SCHUBERT, S., NEUBERT, P., GARG, S., et al. "Visual place recognition: A tutorial", *arXiv preprint arXiv:2303.03281*, 2023.

[45] KEETHA, N., MISHRA, A., KARHADE, J., et al. "AnyLoc: Towards Universal Visual Place Recognition", *IEEE Robotics and Automation Letters*, v. 9, n. 2, pp. 1286–1295, 2024. doi: 10.1109/LRA.2023.3343602.

[46] MILFORD, M., WYETH, G. "Hippocampal models for simultaneous localisation and mapping on an autonomous robot". In: *Proceedings of the 2003 Australasian Conference on Robotics and Automation*, pp. 1–10. Australian Robotics and Automation Association, 2003.

[47] MILFORD, M. J., WYETH, G. F. "Mapping a suburb with a single camera using a biologically inspired SLAM system", *IEEE Transactions on Robotics*, v. 24, n. 5, pp. 1038–1053, 2008.

[48] MILFORD, M., WYETH, G. "Persistent navigation and mapping using a biologically inspired SLAM system", *International Journal of Robotics Research*, v. 29, n. 9, pp. 1131–1153, 2010.

[49] SILVEIRA, L., GUTH, F., DREWS-JR, P., et al. "An open-source bio-inspired solution to underwater SLAM", *IFAC-PapersOnLine*, v. 48, pp. 212–217, 2015.

[50] GUTH, F. A., SILVEIRA, L., AMARAL, M., et al. "Underwater Visual 3D SLAM Using a Bio-inspired System". In: *Symposium on Computing and Automation for Offshore Shipbuilding*. IEEE, 2013. doi: 10.1109/SCASOS.2013.5123.

[51] YIN, P., ABUDUWEILI, A., ZHAO, S., et al. "BioSLAM: A Bioinspired Lifelong Memory System for General Place Recognition", *IEEE Transactions on Robotics*, v. 39, n. 6, pp. 4855–4865, 2023. doi: 10.1109/TRO.2023.3306615.

[52] GUO, J., CHENG, M., REN, J., et al. "A Bio-inspired SLAM System for a Legged Robot". In: *IEEE International Conference on Industrial Electronics and Applications*. IEEE, 2022. doi: 10.1109/ICIEA.2022.10006322.

[53] SILVEIRA, L. *Sistema Bioinspirado para Mapeamento e Localização de Robôs Móveis em Ambientes Subaquáticos*. Tese de Mestrado, 2015.

[54] BAY, H., TUYTELAARS, T., VAN GOOL, L. "Surf: Speeded up robust features", *Lecture notes in computer science*, v. 3951, pp. 404–417, 2006.

[55] HU, M.-K. "Visual pattern recognition by moment invariants", *IRE transactions on information theory*, v. 8, n. 2, pp. 179–187, 1962.

[56] ARTHUR, D., VASSILVITSKII, S. "K-means++ the advantages of careful seeding". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035, 2007.

[57] STRINGER, S., TRAPPENBERG, T., ROLLS, E., et al. "Self-organizing continuous attractor networks and path integration: one-dimensional models of head direction cells", *Network: Computation in Neural Systems*, v. 13, n. 2, pp. 217–242, 2002.

[58] GIOCOMO, L. M., MOSER, M.-B., MOSER, E. I. "Computational models of grid cells", *Neuron*, v. 71, n. 4, pp. 589–603, 2011.

[59] YARTSEV, M. M., ULANOVSKY, N. "Representation of three-dimensional space in the hippocampus of flying bats", *Science*, v. 340, n. 6130, pp. 367–372, 2013.

[60] ROLLS, E. T. *Cerebral cortex: principles of operation*. Oxford University Press, 2016.

[61] HAWKINS, J., AHMAD, S. "Why neurons have thousands of synapses, a theory of sequence memory in neocortex", *Frontiers in neural circuits*, v. 10, pp. 174222, 2016.

[62] SÜNDERHAUF, N., SHIRAZI, S., DAYOUB, F., et al. "On the performance of convnet features for place recognition". In: *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 4297–4304. IEEE, 2015.

[63] BINGHAM, E., MANNILA, H. "Random projection in dimensionality reduction: applications to image and text data". In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 245–250, 2001.

[64] JOHNSON, W. B., LINDENSTRAUSS, J., OTHERS. "Extensions of Lipschitz mappings into a Hilbert space", *Contemporary mathematics*, v. 26, n. 189-206, pp. 1, 1984.

[65] SÜNDERHAUF, N., SHIRAZI, S., JACOBSON, A., et al. "Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free", *Robotics: Science and Systems XI*, pp. 1–10, 2015.

[66] DASGUPTA, S. "Experiments with random projection", *arXiv preprint arXiv:1301.3849*, 2013.

[67] NEUBERT, P., SCHUBERT, S., PROTZEL, P. "A neurologically inspired sequence processing model for mobile robot place recognition", *IEEE Robotics and Automation Letters*, v. 4, n. 4, pp. 3200–3207, 2019.

[68] LI, P., HASTIE, T. J., CHURCH, K. W. "Improving random projections using marginal information". In: *International Conference on Computational Learning Theory*, pp. 635–649. Springer, 2006.

[69] NEUBERT, P., AHMAD, S., PROTZEL, P. "A sequence-based neuronal model for mobile robot localization". In: *KI 2018: Advances in Artificial Intelligence: 41st German Conference on AI, Berlin, Germany, September 24–28, 2018, Proceedings 41*, pp. 117–130. Springer, 2018.

[70] HAWKINS, J., AHMAD, S., PURDY, S., et al. "Biological and Machine Intelligence (BAMI)", Initial online release 0.4, 2016. Disponível em: <https://numenta.com/resources/biological-and-machine-intelligence/>.

[71] LUPTON, T., SUKKARIEH, S. "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions", *IEEE Transactions on Robotics*, v. 28, n. 1, pp. 61–76, 2011.

[72] FORSTER, C., CARLONE, L., DELLAERT, F., et al. "On-manifold preintegration for real-time visual–inertial odometry", *IEEE Transactions on Robotics*, v. 33, n. 1, pp. 1–21, 2016.

[73] LOWRY, S., SÜNDERHAUF, N., NEWMAN, P., et al. "Visual place recognition: A survey", *ieee transactions on robotics*, v. 32, n. 1, pp. 1–19, 2015.

[74] MUR-ARTAL, R., TARDÓS, J. D. "Fast relocalisation and loop closing in keyframe-based SLAM". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 846–853. IEEE, 2014.

[75] HORN, B. K. "Closed-form solution of absolute orientation using unit quaternions", *Journal of the optical society of America A*, v. 4, n. 4, pp. 629–642, 1987.

[76] ENGEL, J., KOLTUN, V., CREMERS, D. "Direct sparse odometry", *IEEE transactions on pattern analysis and machine intelligence*, v. 40, n. 3, pp. 611–625, 2017.

[77] ZUBIZARRETA, J., AGUINAGA, I., MONTIEL, J. M. M. "Direct sparse mapping", *IEEE Transactions on Robotics*, v. 36, n. 4, pp. 1363–1370, 2020.

[78] HAWKINS, J., AHMAD, S., PURDY, S., et al. "Biological and Machine Intelligence (BaMI)". https://www.numenta.com/resources/htm/biological-and-machine-intelligence/, 2016. Digital book, initial online release 0.4; updated editions available from Numenta.

[79] HAWKINS, J. *A Thousand Brains: A New Theory of Intelligence*. New York, NY, Basic Books, 2021. ISBN: 9781541675810.

[80] AHMAD, S., HAWKINS, J. "Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory", *arXiv preprint arXiv:1503.07469*, 2015. Disponível em: <https://arxiv.org/abs/1503.07469>.

Appendix A

MATLAB Script for Visual Place Recognition Stress Test

```
close all
clearvars
warning('off', 'MATLAB:MKDIR:DirectoryExists');

% Define the desired experiment
% Available options:
% 'robotarium_ratslam', 'robotarium_neoslam', 'corridor',
% 'outdoor_afternoon', 'outdoor_afternoon_ratslam', 'irataus',
% 'loop_3x_caverna02', 'loop_3x_elipse'
exp = 'EXP_CNNShuffleNetV2x1_0_recorded'; % Change as needed

filesSaved = 0;
plotConfig = 0;
ratslam_var = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Configuration based on selected experiment
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
switch exp
    case 'robotarium_ratslam'
        file = fullfile('experiments',
                        'robotarium_ratslam',
                        'ratslam_robotarium_features.bag');
        offset1 = 10;
        offset2 = 20;
        theta = -58;
        figure2_process = 0;
        ratslam_var = 1;
    case 'robotarium_neoslam'
        file = fullfile('experiments',
                        'robotarium_neoslam',
```

```
                        '_2023-09-23-18-38-24_0.bag');
        offset1 = 10;
        offset2 = 20;
        theta = -58;
        figure2_process = 1;
        ratslam_var = 0;
    case 'corridor'
        file = fullfile('experiments',
                        'corridor',
                        '_2023-04-25-13-09-50_0.bag');
        offset1 = 1;
        offset2 = 1;
        theta = 0;
        figure2_process = 0;
        ratslam_var = 0;
    case 'outdoor_afternoon'
        file = fullfile('experiments',
                        'outdoor_afternoon',
                        '_2023-09-29-10-10-04_0.bag');
        offset1 = 1;
        offset2 = 1;
        theta = 60;
        figure2_process = 1;
        ratslam_var = 0;
    case 'outdoor_afternoon_ratslam'
        file = fullfile('experiments',
                        'outdoor_afternoon_ratslam',
                        'ratslam_outdoor_features.bag');
        offset1 = 1;
        offset2 = 1;
        theta = 60;
        figure2_process = 0;
        ratslam_var = 1;
    case 'irataus'
        file = fullfile('experiments',
                        'irataus',
                        'irataus_features_01.bag');
        offset1 = 10;
        offset2 = 20;
        theta = -58;
        figure2_process = 1;
        ratslam_var = 0;
    case 'EXP_CNNAlexNetConv3_recorded'
        file = fullfile('experiments',
                        'underwater_cave',
                        'EXP_CNNAlexNetConv3_recorded.bag');
        offset1 = 1;
```

```
79          offset2 = 1;
80          theta = 60;
81          figure2_process = 1;
82          filesSaved = 1;
83          plotConfig = 1;
84          ratslam_var = 1;
85      case 'EXP_CNNShuffleNetV2x1_0_recorded'
86          file = fullfile('experiments',
87                          'underwater_cave',
88                          'EXP_CNNShuffleNetV2x1_0_recorded.bag');
89          offset1 = 1;
90          offset2 = 1;
91          theta = 60;
92          figure2_process = 1;
93          filesSaved = 1;
94          plotConfig = 1;
95          ratslam_var = 1;
96      case 'EXP_X_ODOM_recorded'
97          file = fullfile('experiments',
98                          'underwater_cave',
99                          'EXP_X_ODOM_recorded.bag');
100         offset1 = 1;
101         offset2 = 1;
102         theta = 60;
103         figure2_process = 1;
104         filesSaved = 1;
105         plotConfig = 1;
106         ratslam_var = 1;
107     otherwise
108         warning('Experiment not recognized. No plot will be
                created.')
109         return
110 end
111
112 % Upload the ROS bag file
113 bag = rosbag(file);
114 bag1 = rosbag("EXP_X_ODOM_recorded.bag");
115 bag2 = rosbag("EXP_CNNShuffleNetV2x1_0_recorded.bag");
116
117
118 % Create directory to save output files
119 outputDir = fullfile(pwd,'experiments',exp,'outputFiles');
120 mkdir(outputDir);
121
122 % Get the topics available in the bag
123 available_topics = bag.AvailableTopics;
124 topic_names = available_topics.Properties.RowNames;
```

```
125 disp(topic_names);
126 msg_type = available_topics.MessageType;
127 disp(msg_type);
128
129 % Add the path where the 'evaluateSim' function is located
130 addpath '/home/plinio/Desktop/neoslam_ws/scripts/matlab/utilFiles'
131
132
133 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
134 % Similarity Matrix Processing
135 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
136  if figure2_process == 1
137      if any(strcmp(topic_names, '/feats_htm'))
138          data_feats_htm = select(bag, 'Topic', '/feats_htm');
139          msg_htm = readMessages(data_feats_htm, 'DataFormat',
                 'struct');
140
141          if ~isempty(msg_htm)
142              D1 = cellfun(@(m) find(m.Data), msg_htm,
                     'UniformOutput', false);
143              S = evaluateSim(D1(offset1:size(msg_htm,1)-offset2),
                     D1(offset1:size(msg_htm,1)-offset2), 'wincell');
144
145              figure(2)
146              s = imagesc(S);
147              colorbar
148              title('Similarity Matrix - CNN ShuffleNetV2 x1.0')
149              xlabel('Query images')
150              ylabel('Database images')
151
152              % Save the image in high resolution (300 DPI)
153             figname = fullfile(outputDir, strcat(exp,
                    '_Similarity_Matrix_ShuffleNetV2_300DPI.png'));
154             print(figname, '-dpng', '-r300');  % Export as PNG at
                    300 DPI
155          else
156              warning('The topic /feats_htm contains no messages.');
157          end
158      else
159          warning('Topic /feats_htm not found in topic list.');
160      end
161  end
162
163
164
165 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
166 % /LocalView/Template Topic Processing (View Cells with Loop
        Closure)
167 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
168 if any(strcmp(topic_names, '/LocalView/Template')) && ratslam_var
        == 1
169     % Select the topic and read the messages
170     bSel_vt = select(bag, 'Topic', '/LocalView/Template');
171     msg_vt = readMessages(bSel_vt, 'DataFormat', 'struct');
172     id = cellfun(@(m) double(m.CurrentId), msg_vt);  % IDs das
            View Cells
173
174     % Identify Loop Closures
175     [unique_ids, ~, ic] = unique(id);  % Unique IDs and Occurrence
            Index
176     counts = accumarray(ic, 1);  % Count how many times each View
            Cell was created
177     loop_closure_ids = unique_ids(counts > 1);  % View Cells with
            Loop Closure
178
179     % Display the total number of View Cells created
180     total_view_cells = max(id);  % Total number of View Cells
            created
181     fprintf('Total number of View Cells created: %d\n',
            total_view_cells);
182
183     % Display the number of Loop Closures detected
184     num_loop_closures = length(loop_closure_ids);  % Number of
            Loop Closures detected
185     fprintf('Number of Loop Closures detected: %d\n',
            num_loop_closures);
186
187     % Create and open .txt file to save Loop Closure occurrences
188     output_file = fullfile(outputDir, strcat(exp,
            '_loop_closures.txt'));
189     fileID = fopen(output_file, 'w');
190
191     % Verify that the file was created successfully
192     if fileID == -1
193         warning('Could not create file to save Loop Closure
                data.');
194     else
195         fprintf(fileID, 'Total number of View Cells created:
                %d\n', total_view_cells);
196         fprintf(fileID, 'Number of Loop Closures detected: %d\n',
                num_loop_closures);
197
```

```
198         % Save View Cells that correspond to Loop Closure to .txt
                file
199         if ~isempty(loop_closure_ids)
200             fprintf(fileID, 'View Cells com Loop Closure
                    detectadas:\n');
201
202             % Save Loop Closure occurrences
203             for i = 1:length(loop_closure_ids)
204                 lc_id = loop_closure_ids(i);
205                 lc_occurrences = find(id == lc_id);  % Indexes
                        where Loop Closure occurred
206                 fprintf(fileID, 'View Cell ID %d occurs in the
                        following positions (indices):\n', lc_id);
207                 % Save comma-separated indexes
208                 fprintf(fileID, '%s\n', strjoin(arrayfun(@num2str,
                        lc_occurrences, 'UniformOutput', false), ', '));
209             end
210         else
211             fprintf(fileID, 'No View Cell with Loop Closure
                    detected.\n');
212         end
213
214     % Close the file after saving data
215     fclose(fileID);
216 end
217
218     % Plot View Cells with Loop Closures Highlighted
219     figure(3)
220     hold on;
221
222     % Plot all View Cells in blue
223     h1 = plot(id, 'bx', 'MarkerSize', 8, 'LineWidth', 1.5);  %
            Store handle without displaying multiple captions
224
225     % Highlight the View Cells that are Loop Closures in red
226     if ~isempty(loop_closure_ids)
227         h2 = plot(find(ismember(id, loop_closure_ids)),
                id(ismember(id, loop_closure_ids)), 'ro', 'MarkerSize',
                8, 'LineWidth', 1.5);  % Storing the handle for Loop
                Closures
228     end
229
230     % Add Legend with Count of View Cells and Loop Closures
231     legend([h1, h2], {sprintf('View Cells created: %d',
            total_view_cells), sprintf('Loop Closure detected: %d',
            num_loop_closures)}, 'Location', 'northeast', 'FontSize',
            8);
```

```
232
233     hold off;
234
235     grid on;
236     grid minor;
237     title('View Cells - NeoSLAM with CNN ShuffleNetV2 x1.0
            integrated', 'FontSize', 12);
238     xlabel('Query images', 'FontSize', 10);
239     ylabel('View Cell Id', 'FontSize', 10);
240
241     % Plot setup and saving
242     if plotConfig == 1
243         % Set the image size and resolution to 300 DPI
244         width = 3.5;  % Recommended width for a column in IEEE
245         height = 3;   % Proportional height
246
247         set(gcf, 'PaperUnits', 'inches');
248         set(gcf, 'PaperPosition', [0 0 width height]);  % Set the
              figure size
249
250         figname31 = fullfile(outputDir,
              strcat(exp,'_vc_loop_closure.png'));
251         figname32 = fullfile(outputDir,
              strcat(exp,'_vc_loop_closure.eps'));
252
253         % Save PNG at 300 DPI
254         print(figname31, '-dpng', '-r300');
255
256         % Save EPS to Vector Graphics
257         print(figname32, '-depsc', '-r300');
258
259         if filesSaved == 1
260             exportgraphics(gcf, figname31);
261             exportgraphics(gcf, figname32);
262         end
263     end
264 end
265
266
267 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
268 % Processing the Topic /ExperienceMap/Map (Map Experience) with
      two bags
269 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
270
271 % Read data from two bags
272 if any(strcmp(topic_names, '/ExperienceMap/Map'))
273     % Select the topics for each bag
```

```
274     data_em1 = select(bag1, 'Topic', '/ExperienceMap/Map');
275     data_em2 = select(bag2, 'Topic', '/ExperienceMap/Map');
276 
277     % Read the messages in each bag
278     msg1 = readMessages(data_em1, 'DataFormat', 'struct');
279     msg2 = readMessages(data_em2, 'DataFormat', 'struct');
280 
281     % Check if messages are empty
282     if isempty(msg1)
283         warning('No messages found in /ExperienceMap/Map for
                bag1.')
284     end
285     if isempty(msg2)
286         warning('No messages found in /ExperienceMap/Map for
                bag2.')
287     end
288 
289     % If there are messages in both bags
290     if ~isempty(msg1) && ~isempty(msg2)
291         % Process the last message of each bag
292         em1 = msg1{end};
293         em2 = msg2{end};
294 
295         % Initialize variables to store poses and links
296         pose_em1 = [];
297         pose_em2 = [];
298         links1 = [];
299         links2 = [];
300 
301         % Process the first bag
302         if ~isempty(em1.Node)
303             x1 = em1.Node(:);
304             y0_1 = double(vertcat(x1.Id));
305             y1_1 = vertcat(x1.Pose)';
306             z1_1 = vertcat(y1_1.Position);
307             z2_1 = vertcat(y1_1.Orientation);
308             w1 = vertcat([z1_1.X; z1_1.Y; z1_1.Z; z2_1.X; z2_1.Y;
                    z2_1.Z; z2_1.W])';
309             pose_em1 = [y0_1 w1];
310 
311             x_edges1 = em1.Edge(:);
312             y0_1 = vertcat(x_edges1.Id);
313             y1_1 = vertcat(x_edges1.SourceId);
314             y2_1 = vertcat(x_edges1.DestinationId);
315             links1 = [y0_1 y1_1 y2_1];
316         else
```

```
317             warning('No nodes found in /ExperienceMap/Map for
                    bag1.')
318         end
319
320         % Process the second bag
321         if ~isempty(em2.Node)
322             x2 = em2.Node(:);
323             y0_2 = double(vertcat(x2.Id));
324             y1_2 = vertcat(x2.Pose)';
325             z1_2 = vertcat(y1_2.Position);
326             z2_2 = vertcat(y1_2.Orientation);
327             w2 = vertcat([z1_2.X; z1_2.Y; z1_2.Z; z2_2.X; z2_2.Y;
                    z2_2.Z; z2_2.W])';
328             pose_em2 = [y0_2 w2];
329
330             x_edges2 = em2.Edge(:);
331             y0_2 = vertcat(x_edges2.Id);
332             y1_2 = vertcat(x_edges2.SourceId);
333             y2_2 = vertcat(x_edges2.DestinationId);
334             links2 = [y0_2 y1_2 y2_2];
335         else
336             warning('No nodes found in /ExperienceMap/Map for
                    bag2.')
337         end
338
339         % Plot the pose data of both bags
340         f4 = figure(4);
341         hold on;
342
343         markersize = 5;
344         linewidth = 2;
345         draw_links = 1;
346
347         % Plot the first bag (Odometry) in blue, without including
                it in the legend
348         plot(pose_em1(:,2), pose_em1(:,3), 'b-', 'MarkerSize',
                markersize, 'HandleVisibility', 'off');  % Hidden from
                caption
349
350         % Add markers at the start and end of the Odometry route
351         plot(pose_em1(1,2), pose_em1(1,3), 'bo', 'MarkerSize',
                markersize*3, 'LineWidth', linewidth*2);  % Start marker
352         plot(pose_em1(end,2), pose_em1(end,3), 'bx', 'MarkerSize',
                markersize*3, 'LineWidth', linewidth*2);  % End marker
353
354         % Plot the second bag (Underwater NeoSLAM) in red, without
                including it in the legend
```

```
355         plot(pose_em2(:,2), pose_em2(:,3), 'r--', 'MarkerSize',
                markersize, 'HandleVisibility', 'off');  % Hidden from
                caption
356
357         % Add markers at the start and end of the NeoSLAM path
358         plot(pose_em2(1,2), pose_em2(1,3), 'ro', 'MarkerSize',
                markersize*3, 'LineWidth', linewidth*2);  % Start marker
359         plot(pose_em2(end,2), pose_em2(end,3), 'rx', 'MarkerSize',
                markersize*3, 'LineWidth', linewidth*2);  % End marker
360
361         % Create dummy objects for the caption
362         h1 = plot(NaN, NaN, 'b-', 'MarkerSize', markersize,
                'LineWidth', linewidth);  % Odometry Line
363         h2 = plot(NaN, NaN, 'r--', 'MarkerSize', markersize,
                'LineWidth', linewidth);  % NeoSLAM Line
364
365         % Create markers for start and end of Odometry path
366         h3 = plot(NaN, NaN, 'bo', 'MarkerSize', markersize*3,
                'LineWidth', linewidth*2);  % Home of Odometry
367         h4 = plot(NaN, NaN, 'bx', 'MarkerSize', markersize*3,
                'LineWidth', linewidth*2);  % End of Odometry
368
369         % Create markers for the start and end of the NeoSLAM path
370         h5 = plot(NaN, NaN, 'ro', 'MarkerSize', markersize*3,
                'LineWidth', linewidth*2);  % NeoSLAM Start
371         h6 = plot(NaN, NaN, 'rx', 'MarkerSize', markersize*3,
                'LineWidth', linewidth*2);  % End of NeoSLAM
372
373         % Add subtitle with start and end markers
374         legend([h1, h2], {'Odometry', 'NeoSLAM'}, ...
375                'AutoUpdate', 'off', 'Location', 'northeast',
                    'FontSize', 8);
376
377         % Disable automatic subtitle update (prevents new
                additions)
378         set(legend, 'AutoUpdate', 'off');
379
380         title('Experience Map - NeoSLAM with CNN ShuffleNetV2 x1.0
                integrated', 'FontSize', 14);
381         xlabel('x(m)', 'FontSize', 12);
382         ylabel('y(m)', 'FontSize', 12);
383         axis equal;  % To maintain equal proportions between the
                axes
384         grid on;
385         grid minor;  % Subtle grid for clarity
386
387         hold off;
```

```
388
389         % Plot the links of the first bag
390         if draw_links == 1 && ~isempty(links1)
391             nl1 = size(links1, 1);
392             for k = 1:nl1
393                 sn1 = links1(k, 2) + 1;
394                 dn1 = links1(k, 3) + 1;
395                 if sn1 > size(pose_em1, 1) || dn1 > size(pose_em1,
                        1)
396                     warning('Bag1 link indices exceed the number
                            of poses.');
397                     continue;
398                 end
399                 line([pose_em1(sn1, 2) pose_em1(dn1, 2)],
                        [pose_em1(sn1, 3) pose_em1(dn1, 3)], 'Color',
                        'blue');
400             end
401         end
402
403         % Plot the links of the second bag
404         if draw_links == 1 && ~isempty(links2)
405             nl2 = size(links2, 1);
406             for k = 1:nl2
407                 sn2 = links2(k, 2) + 1;
408                 dn2 = links2(k, 3) + 1;
409                 if sn2 > size(pose_em2, 1) || dn2 > size(pose_em2,
                        1)
410                     warning('bag2 link indices exceed number of
                            poses.');
411                     continue;
412                 end
413                 line([pose_em2(sn2, 2) pose_em2(dn2, 2)],
                        [pose_em2(sn2, 3) pose_em2(dn2, 3)], 'Color',
                        'red');
414             end
415         end
416
417         hold off;
418
419         % Plot setup and saving (same as original)
420         if plotConfig == 1
421             width = 3;
422             height = 3;
423             alw = 0.75;
424             fsz = 9;
425             lw = 1.5;
426             msz = 8;
```

```
427
428             set(gcf, 'Position', [100, 100, width*100,
                    height*100]);
429             set(findall(gcf, '-property', 'FontSize'), 'FontSize',
                    fsz);
430             set(findall(gcf, '-property', 'LineWidth'),
                    'LineWidth', alw);
431
432             figname41 = fullfile(outputDir, strcat(exp,'_em.jpg'));
433             figname42 = fullfile(outputDir,
                    strcat(exp,'_em_eps.eps'));
434
435             % Save PNG at 300 DPI
436             print(figname41, '-dpng', '-r300');
437
438             % Save EPS to Vector Graphics
439             print(figname42, '-depsc', '-r300');
440
441             if filesSaved == 1
442                 exportgraphics(gcf, figname41);
443                 exportgraphics(gcf, figname42);
444             end
445         end
446     end
447 end
448
449
450
451 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
452 % RatSlam Related Variables Processing
453 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
454 if any(strcmp(topic_names, '/LocalView/Template')) && ratslam_var
        == 1
455     bSel_vt = select(bag, 'Topic', '/LocalView/Template');
456     msg_vt = readMessages(bSel_vt, 'DataFormat', 'struct');
457     id = cellfun(@(m) double(m.CurrentId), msg_vt); %Extracts the
            current ID from the "View Cells" of each message in the
            topic
458
459     % Confusion Matrix Processing with RatSlam
460     features = cellfun(@(m) double(m.Feature), msg_vt,
            'UniformOutput', false); %Extracts the visual features
            associated with each message
461     feature_max = max(cell2mat(features)); %Calculates the maximum
            feature across all data for normalization
462     features_values = length(features);
```

```
463     ratslam_matrix = eye(features_values); %Creates an identity
            matrix of size equal to the number of features
464     for i = 1:(features_values-1)
465         aux = cell2mat(features(i+1));
466         aux_size = length(aux);
467         for j = 1:i
468             if j <= aux_size
469                 ratslam_matrix(j,i+1) = (feature_max -
                        aux(j))/feature_max;
470             end
471         end
472     end
473
474     figure(6);
475     imagesc(ratslam_matrix);
476     colorbar;
477     title('Confusion Matrix');
478     xlabel('Query images');
479     ylabel('Database images');
480
481     % Save figure 6 in high resolution
482     figname6 = fullfile(outputDir, strcat(exp,
            '_confusion_matrix_ratslam.png'));
483     print(figname6, '-dpng', '-r300');  % Export as PNG at 300 DPI
484
485     vc_created = max(id) + 2; %Calculates the total number of
            "View Cells" created by adding a buffer of two IDs
486     ratslam_vc_matrix = eye(vc_created); %Creates an identity
            matrix for the View Cells
487     aux_size_prev = 0;
488     for i = 1:features_values
489         aux = cell2mat(features(i));
490         aux_size = length(aux);
491
492         if aux_size == aux_size_prev
493             continue
494         end
495
496         for j = 1:aux_size
497             ratslam_vc_matrix(j, aux_size + 1) = (feature_max -
                    aux(j))/feature_max;
498         end
499
500         aux_size_prev = aux_size;
501     end
502
503     figure(7);
```

```
504     imagesc((ratslam_vc_matrix + ratslam_vc_matrix') / 2); %Plots
            the similarity matrix between View Cells, normalizing it
            with the transpose
505     colorbar;
506     title('View Cells Similarity Matrix');
507     xlabel('Query View Cells');
508     ylabel('Database View Cells');
509
510     % Save figure 7 in high resolution
511     figname7 = fullfile(outputDir, strcat(exp,
            '_view_cells_similarity_matrix.png'));
512     print(figname7, '-dpng', '-r300');  % Export as PNG at 300 DPI
513
514     figure(8);
515     GT = createGT(vc_created, 0, 20, [16:289, 412:664, 773:1018,
            1360:2069, 2076:2742, 2750:3370]); %Generates a Ground
            Truth (GT) for mapping validation and loop closure
516     imagesc(GT); %Plots the Ground Truth matrix, indicating the
            expected loop closures
517     colorbar;
518     title('Ground Truth - NeoSLAM with CNN ShuffleNetV2 x1.0
            integrated');
519     xlabel('Query View Cells');
520     ylabel('Database View Cells');
521
522     % Save figure 8 in high resolution
523     figname8 = fullfile(outputDir, strcat(exp,
            '_ground_truth.png'));
524     print(figname8, '-dpng', '-r300');  % Export as PNG at 300 DPI
525
526     figure(9);
527     [P, R, F1] = createPR((ratslam_vc_matrix + ratslam_vc_matrix')
            / 2, GT); %Calculates Precision (P), Recall (R), and
            F1-Score metrics using the similarity matrix and Ground
            Truth
528
529     % Plot the Precision-Recall curve
530     plot(R, P); % Plot the Precision vs Recall curve
531     title('Precision-Recall Curve - NeoSLAM with CNN ShuffleNetV2
            x1.0 integrated');
532     xlabel('Recall');
533     ylabel('Precision');
534
535     % Add legend with P, R and F1 values
536     legend_text = sprintf('Precision: %.3f, Recall: %.3f,
            F1-Score: %.3f', mean(P), mean(R), mean(F1));
```

```
537     legend(legend_text, 'Location', 'northeast', 'FontSize', 8); %
            Displays caption at top right
538
539     % Save figure 9 in high resolution
540     figname9 = fullfile(outputDir, strcat(exp,
            '_precision_recall_curve.png'));
541     print(figname9, '-dpng', '-r300');  % Export as PNG at 300 DPI
542 end
```

Listing A.1: bagROS Plot Underwater Cave Experiment for CNN ShuffleNetV2 x1.0