UNIVERSIDADE FEDERAL DO MARANHÃO

UNIVERSIDADE FEDERAL DO PIAUÍ

Doutorado em Ciência da Computação Associação

UFMA/UFPI

Matheus Chaves Menezes

Biologically inspired approaches to building spatial maps for spatial navigation and learning

Advisor: Prof. Dr. Alexandre César Muniz de Oliveira

Co-advisor: Prof. Dr. Sen Cheng

São Luís - MA
July, 2023

Matheus Chaves Menezes

Biologically inspired approaches to building spatial maps for spatial navigation and learning

TESE DE DOUTORADO

Thesis presented as a partial requirement for obtaining the title of Doutor em Ciência da Computação to the Doutorado em Ciência da Computação Associação UFMA/UFPI.

Advisor: Prof. Dr. Alexandre César Muniz de Oliveira
Co-advisor: Prof. Dr. Sen Cheng

São Luís - MA
July, 2023

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Menezes, Matheus Chaves.
   Biologically inspired approaches to building spatial maps for spatial navigation and learning / Matheus Chaves Menezes. - 2023.
   107 p.

   Coorientador(a): Sen Cheng.
   Orientador(a): Alexandre César Muniz de Oliveira.
   Tese (Doutorado) - Programa de Pós-graduação Doutorado em Ciência da Computação - Associação UFMA/UFPI, Universidade Federal do Maranhão, São Luís, 2023.

   1. Cognitive maps. 2. Latent learning. 3. RatSLAM. 4. Spatial navigation. 5. Successor representation. I. Cheng, Sen. II. Oliveira, Alexandre César Muniz de. III. Título.

Matheus Chaves Menezes

Biologically inspired approaches to building spatial maps for spatial navigation and learning

This Tese de Doutorado was evaluated and approved by an examination committee composed of the following members:

Prof. Dr. Alexandre César Muniz de Oliveira
Advisor
Universidade Federal do Maranhão

Prof. Dr. Sen Cheng
Co-advisor
Ruhr University Bochum

Prof. Dr. Dante Augusto Couto Barone
External Examiner
Universidade Federal do Rio Grande do Sul

Prof. Dr. Edison Pignaton De Freitas
External Examiner
Universidade Federal do Rio Grande do Sul

Prof. Dr. Areolino De Almeida Neto
Internal Examiner
Universidade Federal do Maranhão

Prof. Dr. Luciano Reis Coutinho
Internal Examiner
Universidade Federal do Maranhão

We certify that this is the original and final version of the Tese de Doutorado that was deemed suitable for obtaining the title of Doutor em Ciência da Computação.

Prof. Dr. Alexandre César Muniz de Oliveira
Advisor

Prof. Dr. André Castelo Branco Soares
Coordinator of the Doutorado em Ciência da Computação Associação UFMA/UFPI.

São Luís - MA, July 27th of 2023

In memory of my dear father, whose teachings, dedication, and unconditional love have given me strength on this journey.

Acknowledgements

First, I would like to thank my parents, Coracy Chaves and Jorge Menezes, for always supporting me in everything I dreamed of. They always fought for my future and inspired me to learn along the way. My love for you will be eternal.

To my life partner, Amanda Paixão, I thank you for all the love and support since we met. Your dedication and patience were invaluable during my doctorate and part of this work is due to you.

To my family and my siblings who always believed in me. In particular, I would like to thank my sister, Maurícia Menezes, who dedicated part of her life to helping me with my studies. I thank my sister, Samantha Menezes, for all her love and care for me and for our father. I hope that one day I can become a good example for you.

To my advisors, Alexandre de Oliveira, Paulo Ribeiro, and Sen Cheng, for all the knowledge and learning transmitted in recent years. Alexandre believed in me since the first period of college, and I am grateful for our friendship beyond teacher-student. Paulo has always encouraged me to pursue my career, academic or otherwise. Sen gave me opportunities to explore new areas of knowledge.

To all my friends who have been by my side. Especially Marcos Oliveira and Gilvan Tavares for everything we've experienced in these last 11 years of friendship. To my friend Ítalo Francyles who, like me, took on the challenge of a doctorate with a lot of will and courage. To my friends Mellanie Trinta and Julliane César for all their support and friendship. To my friend Moisés Rocha who is a great person and inspires me greatly.

To all DCCMAPI professors and staff, especially to Anselmo Paiva and Salles Neto, for teaching and collaboration.

To everyone who, in one way or another, encouraged or helped me.

This work was carried out with the support of the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) - Financing Code 001.

“In science, there are no shortcuts to truth.”

Carl Sagan.

Resumo

Navegar por ambientes desconhecidos e buscar por recursos, como comida e água, é fundamental para a sobrevivência de muitos animais, incluindo os seres humanos. Por quase um século, pesquisas em neurociência comportamental e cognitiva apoiam a existência dos mapas cognitivos, sendo usados por animais para navegar no espaço. Mapas cognitivos permitem que animais realizem tarefas complexas, incluindo a aquisição de um mapa global a partir de ambientes distintos uma vez que as conexões entre eles são estabelecidas. Ademais, pesquisas apontam que a construção de um mapa cognitivo prévio à participação em tarefas recompensadas pode acelerar o processo de aprendizado, conforme evidenciado por experimentos de aprendizado latente. No entanto, os fatores específicos que contribuem para as diferenças observadas na velocidade de aprendizado, influenciadas por projetos experimentais e estratégias de exploração no aprendizado latente, permanecem uma questão em aberto. Esta tese de doutorado propõe novas abordagens computacionais inspiradas em princípios biológicos para a construção de mapas espaciais que facilitem a navegação e aprendizado espacial. O algoritmo de Localização e Mapeamento Simultâneos (SLAM), inspirado no processo de navegação no cérebro de roedores, conhecido como RatSLAM, foi ampliado através do desenvolvimento de uma nova abordagem de fusão de estruturas para lidar com o desafio do mapeamento em múltiplas sessões. O RatSLAM também é integrado como um algoritmo de aprendizado de representação de estado dentro do framework CoBeL-RL, um framework de aprendizado por reforço construído com base em descobertas recentes em neurociência, permitindo que agentes aprendam tarefas espaciais em ambientes desconhecidos. Ao utilizar esse framework, experimentos de aprendizado latente são investigados para obter percepções sobre o impacto dos diferentes projetos experimentais e estratégias de exploração na velocidade de aprendizado. Os resultados evidenciam o êxito do RatSLAM no mapeamento em múltiplas sessões com o uso de conjuntos de dados reais, bem como a habilidade de agentes virtuais de aprender tarefas espaciais em ambientes desconhecidos. Além disso, evidencia-se que os agentes desenvolvem Representações Sucessoras singulares, dependendo dos projetos experimentais específicos, o que oferece uma explicação potencial para as variações na velocidade do aprendizado nos experimentos de aprendizado latente. No geral, esta tese contribui para a robótica e neurociência computacional aprofundando a compreensão dos processos cognitivos envolvidos na navegação espacial e fornecendo percepções práticos para o desenvolvimento de sistemas robóticos mais eficazes e modelos computacionais inspirados em princípios biológicos.

Palavras-chave: Navegação Espacial, Mapas Cognitivos, Aprendizado Latente, Representações Sucessoras, RatSLAM.

Abstract

Navigating unfamiliar spaces and searching for resources, such as food and water, is fundamental for survival in many animals, including humans. For nearly a century, behavioral and cognitive neuroscience research has supported the existence of cognitive maps, which animals employ to navigate spatially. Cognitive maps enable animals to perform complex tasks, including acquiring a global map from distinct contexts once connections are established. Furthermore, studies have revealed that building a cognitive map before engaging in reward-based tasks can enhance learning speed, as evidenced by latent learning experiences. However, the specific factors contributing to the observed differences in learning speed, as influenced by experimental design and exploration strategies in latent learning, remain an open question. This doctoral thesis proposes novel computational approaches inspired by biological principles for building spatial maps to facilitate spatial navigation and learning. The Simultaneous Localization and Mapping (SLAM) algorithm, inspired by the navigation process in rodent brains, known as RatSLAM, has been extended by developing a novel structure merge approach to address the challenge of multisession mapping. RatSLAM is also integrated as a state representation learning algorithm within the CoBeL-RL framework, a reinforcement learning framework built on recent neuroscience findings, enabling agents to learn spatial tasks in unknown environments. By utilizing this framework, latent learning experiments are investigated to gain insights into the impact of different experimental designs and exploration strategies on learning speed. The results demonstrate RatSLAM's successful performance in multisession mapping using real-world datasets and the ability of virtual agents to learn spatial tasks in unfamiliar environments. Additionally, it is shown that agents acquire distinct Successor Representations based on the specific experimental designs, providing a potential explanation for variations in learning speed for latent learning experiments. Overall, this thesis contributes to robotics and computational neuroscience by deepening our understanding of the cognitive processes involved in spatial navigation and providing practical insights for developing more effective robotic systems and computational models inspired by biological principles.

Keywords: Spatial Navigation, Cognitive Maps, Latent Learning, Successor Representation, RatSLAM.

List of Figures

Figure 1 – Spatial related cells found in mammals/rodents brain 16
Figure 2 – RatSLAM architecture. 22
Figure 3 – Interconnections among the components of RatSLAM. 23
Figure 4 – Agent-environment interaction in a Markov decision process. 27
Figure 5 – SR occupancy grid. 33
Figure 6 – The CoBeL-RL architecture. 35
Figure 7 – Tolman's experiments in latent learning. 43
Figure 8 – Condition for the merging procedure between two RatSLAM structures. 49
Figure 9 – Merging partial and loaded RatSLAM maps. 50
Figure 10 – Virtual Tour environment setup. 56
Figure 11 – Lab Tour experiment setup. 57
Figure 12 – The *iRat* experiment setup. 58
Figure 13 – The New College Oxford Experiment Setup. 59
Figure 14 – EM of the counterclockwise lap. 62
Figure 15 – EM of the clockwise lap with counterclockwise parameters. 62
Figure 16 – EM of the *eight* lap with counterclockwise parameters. 62
Figure 17 – Experience Maps (EM) of the Virtual Tour experiment. 64
Figure 18 – ICP comparison for *Virtual Tour* experiment. 64
Figure 19 – Pose Cells Network (PCN) activation in the Virtual Tour experiment. 65
Figure 20 – No shifting association between PCN and LVC. 66
Figure 21 – Experience maps (EM) of the Lab Tour experiment. 67
Figure 22 – ICP comparison for *Lab Tour* experiment. 67
Figure 23 – Experience maps (EM) of the *iRat* experiment. 68
Figure 24 – ICP comparison of EM for *iRat* experiment. 69
Figure 25 – Experience maps (EM) of the *Oxford New College* experiment. . . . 70
Figure 26 – ICP comparison of EM for *Oxford New College* experiment. 70
Figure 27 – Integration of RatSLAM in CoBeL-RL framework. 73
Figure 28 – Possible actions. 74
Figure 29 – Virtual Environment. 76
Figure 30 – Reward learning in Blodgett's maze. 77
Figure 31 – Reward learning in Tolman's maze. 78
Figure 32 – Agent's escape latency in Latent Learning. 79
Figure 33 – Deep Successor Representation learned during Latent Learning. . . 86
Figure 34 – Successor Representation row mapped to environment topology. . . 87
Figure 35 – Q-values in direct and latent learning agents. 88
Figure 36 – Different Designs in Latent Learning. 89

Figure 37 – Deep Successor Representation analysis over design experiments. 90
Figure 38 – Extracted Q-values from SR and ground-truth reward. 91
Figure 39 – Actions of the highest q-values for each state. 91
Figure 40 – Effect of doors in Latent Learning. 92
Figure 41 – Latent Learning in different RL policies. 93

List of Tables

Table 1 – RatSLAM Parameters 27
Table 2 – Parameter of RatSLAM's to each tested environment. 55
Table 3 – Scientific productions related to the proposed methods as the first author. 97
Table 4 – Scientific productions related to the proposed methods and the spatial navigation field as the co-author. 98

List of abbreviations and acronyms

DNN	*Deep Neural Network*
ICP	*Iterative Closest Point*
MB	*Model-based*
MF	*Model-free*
RL	*Reinforcement Learning*
SR	*Successor Representation*

Contents

1 INTRODUCTION . . . 15
1.1 Cognitive Maps and Spatial Cells . . . 15
1.2 Learning to Navigate with Cognitive Maps . . . 16
1.3 Computational Frameworks . . . 17
1.4 Objectives and Contributions . . . 19

2 THEORETICAL FOUNDATION . . . 20
2.1 SLAM . . . 20
2.2 RatSLAM . . . 21
2.2.1 Pose Cells Network . . . 22
2.2.2 Local View Cells . . . 24
2.2.3 Experience Map . . . 25
2.2.4 Role of RatSLAM Parameters . . . 26
2.3 Reinforcement learning . . . 26
2.3.1 Elements of Reinforcement Learning . . . 26
2.3.2 Model-free and Model-based RL . . . 29
2.3.3 The Successor Representation . . . 32
2.4 CoBeL-RL . . . 34
2.4.1 Environment Module . . . 35
2.4.2 Agent Modules . . . 36
2.4.3 Analysis Modules . . . 38
2.5 Summary . . . 38

3 RELATED WORKS . . . 39
3.1 Building Maps with Neuro-Inspired Frameworks . . . 39
3.2 The Multisession Problem for RatSLAM . . . 40
3.3 Learning Cognitive Maps for Flexible Behavior . . . 42
3.4 Experimental Designs in Latent Learning . . . 45
3.5 Summary . . . 47

4 A MULTISESSION APPROACH FOR RATSLAM . . . 48
4.1 Methodology . . . 48
4.1.1 Merging Local View Cells . . . 49
4.1.2 Pose Cells Network Activation . . . 50
4.1.3 Shifting Association between LVC and PCN . . . 51
4.1.4 Merging Experience Maps . . . 51

4.1.5 Merge Algorithm 52
4.2 Experimental Setup 53
4.2.1 *Virtual Tour* experiment 54
4.2.2 *Lab Tour* experiment 55
4.2.3 *iRat* experiment 56
4.2.4 *New College* experiment 57
4.2.5 RatSLAM's parameter regularization 59
4.3 Results 63
4.3.1 *Virtual Tour* Results 63
4.3.2 *Lab Tour* Results 65
4.3.3 *iRat* Results 66
4.3.4 New College Results 68
4.4 Summary 69

5 SPATIAL LEARNING BASED ON COGNITIVE MAPS 72
5.1 CoBeL-RL and RatSLAM Integration 72
5.2 CoBeL-RL and RatSLAM in Rewarding Spatial Task 75
5.3 Latent Learning During Spatial Exploration 77
5.4 Impact of Subtle Differences in Task Design on Latent Learning 81
5.5 Impact of Exploration on Latent Learning 83
5.6 Summary 85

6 DISCUSSION AND CONCLUSION 94
6.1 Enhancing Mapping with RatSLAM 94
6.2 Enhancing Learning with Latent Learning 95
6.3 Scientific Productions 97

BIBLIOGRAPHY 99

1 Introduction

Navigating unfamiliar spaces and searching for resources such as food and water is essential for many animals' survival, including humans. Although the specific mechanisms underlying spatial navigation are complex and unclear, evidence suggests that the process involves two critical components: mapping the environment through exploration and efficiently navigating on that map towards specific goals (POULTER; HARTLEY; LEVER, 2018; HILLS et al., 2015). Nevertheless, how animals navigate has been a recurring research question.

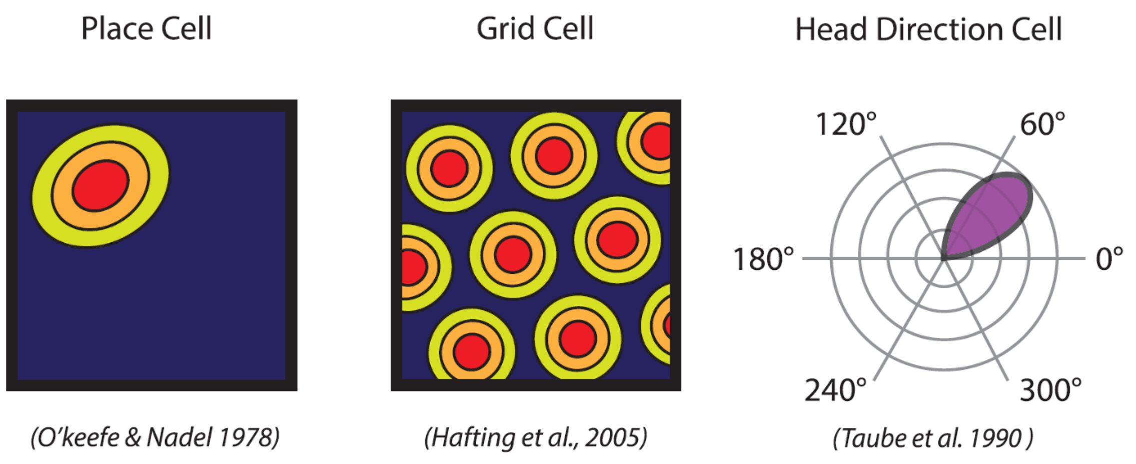
In the mapping phase, spatial-related cells in the hippocampal formation (O'KEEFE; NADEL, 1978; TAUBE; MULLER; RANCK, 1990; HAFTING et al., 2005; BEHRENS et al., 2018) integrate animals' self-motion estimation and environmental cues information (JIN et al., 2020; POULTER; HARTLEY; LEVER, 2018), allowing them to construct a cognitive map of the space (TOLMAN, 1948). This cognitive map serves as one of the foundations for efficient navigation, in which animals can learn and adopt distinct strategies to navigate familiar and unfamiliar places (JIN et al., 2020). One could use stimulus-response (S-R) association, where specific actions are executed in response to stimuli, or use spatial relationships between landmarks and compute navigation paths internally with the cognitive map.

1.1 Cognitive Maps and Spatial Cells

Behavioral experiments on latent learning conducted by Tolman and Honzik (TOLMAN; HONZIK, 1930; TOLMAN, 1948) provided evidence that animals create an internal representation of their environment, the cognitive map, during maze exploration. In this type of learning, groups of animals are pre-exposed to an empty maze for a specific period or until they reach non-rewarded goals. Consequently, even without explicit rewards, rats with prior maze familiarization could locate food more effectively, avoiding dead ends, computing mental detours, and finding environmental shortcuts, outperforming rats with no previous experience but with food available from the start.

The discovery of specific types of brain cells that activate for spatial features of the environment plays a crucial role in spatial navigation and has supported the cognitive map theory. O'Keefe and Dostrovsky discovered "place cells" in the hippocampus that activate when an animal is located in a circumscribed region of space, forming a place field with intense activity at the center of the animal's location (O'KEEFE; NADEL, 1978) (Fig. 1). In addition, Taube and collaborators reported "head direction cells" in several brain regions that activate when the animal's head is rotated in a distinct direction (TAUBE; MULLER;

Figure 1 – Spatial related cells found in mammals/rodents brain



Source: Adapted from (BEHRENS et al., 2018)

RANCK, 1990). Furthermore, grid cells discovered by Hafting et al. (HAFTING et al., 2005) in the entorhinal cortex are activated when the animal's location aligns with a vertex of a hexagonal grid overlaid on the environment, known as a grid field. While pose cells seem to represent an animal's position in space, the periodic activation pattern of grid cells allows for the formation of highly efficient spatial representations. It is believed to be a metric for spatial calculations (HAFTING et al., 2005).

In addition, researches with grid cells have shown that cognitive maps can be adapted and changed to suit complex mapping scenarios, such as building global maps from initially distinct environments. A study by (CARPENTER et al., 2015) demonstrated that initial exposure of rats to multi-compartment environments resulted in grid fields dominated by local compartments. However, with increasing experience, the discontinuities in the grid fields between the compartments gradually reduced, forming a continuous representation spanning both compartments. In further research by (WERNLE et al., 2018), rats mapped two distinct environments separated by a wall, and the grid field formed local representations of each compartment. Removing the wall rapidly adjusted the grid field in the merged location, establishing a coherent global representation of the environment. These findings suggest that grid cells modify their firing patterns to produce a global model of the environment.

1.2 Learning to Navigate with Cognitive Maps

Studies on cognitive maps go beyond their acquisition and explore their role in facilitating efficient spatial navigation and learning. Recent research has investigated the idea of predictive maps, which are inherent in cognitive maps and capture the spatial relationships within an environment (STACHENFELD; BOTVINICK; GERSHMAN, 2017; de

Cothi et al., 2022; BRUNEC; MOMENNEJAD, 2022). Place cells in the brain encode not only an animal's current location but also future locations the animal anticipates visiting from its current location. These relationships are learned based on environmental features during exploration, independently of explicit motivations such as rewards. Subsequently, the predictive relationships of places and future rewards presented in the environment can align to build navigation trajectories, matching Tolman's hypothesis of cognitive maps in latent learning experiments.

Nevertheless, how the predictive maps are learned during the exploration can significantly impact their subsequent use. In a study by Karn e Jr. (1946), they investigated how different types of pre-exposure to a maze affected rats' ability to find subsequent rewards in latent learning. The rats were pre-exposed through diverse experiment designs that included manipulations with and without exploration. Results indicated that the latent learning experiment closely aligned with the rewarded task, i.e., stopping the animals' exploration when finding the empty goal compartment, was more effective in improving learning performance. Furthermore, pre-exposure with exploratory experiences substantially affected rewarded task performance, outperforming groups of animals that did not explore the environment. These findings emphasize the crucial role of exploration during cognitive map acquisition in facilitating navigation toward a goal. However, it is still necessary to elucidate how these different experimental designs affect learning. Our hypothesis asserts that different exploratory designs led to different representations of predictive maps during unrewarded periods, thus impacting subsequent learning performance.

1.3 Computational Frameworks

The study of spatial navigation in animals has enhanced the field's understanding and also driven the development of computational approaches to solve engineering challenges and advance further research through computational models. One notable application is the adoption of neuro-inspired approaches that address the Simultaneous Localization and Mapping (SLAM) problem in mobile robots (ZENO; PATEL; SOBH, 2016; TANG; YAN; TAN, 2018; YU et al., 2019). SLAM involves an agent constructing a map while navigating an unknown environment and determining its position on the map (DURRANT-WHYTE; BAILEY, 2006). RatSLAM, a neuro-inspired solution to solve SLAM, is based on the underlying spatial navigation mechanism in the rodent's brain (MILFORD; WYETH; PRASSER, 2004; MILFORD; WILES; WYETH, 2010; BALL et al., 2013). It builds experience maps by integrating visual and speed information into a conjunctive grid and head direction cells named Pose Cell Networks. However, early versions of RatSLAM faced challenges in handling the complexities of mapping tasks involving multiple sessions.

Reinforcement Learning (RL) has also played a crucial role in investigating how biological agents learn to navigate in space (TESSEREAU et al., 2021; HE et al., 2022; DIEKMANN et al., 2023; de Cothi et al., 2022). This approach involves an agent interacting with an environment, observing states, taking actions, and receiving feedback in the form of rewards or punishments. Through trial and error, the agent learns to make decisions that maximize long-term rewards by adjusting its actions based on past experiences and observed states (SUTTON; BARTO, 2018).

The RL states of the environment are crucial for an agent's decision-making and learning optimal behavior by providing essential contextual information. However, dealing with high-dimensional observations and the absence of compact state representations can introduce complexity. To address this, state representation learning (SRL) is commonly used as a preliminary step (LESORT et al., 2018). SRL aims to discover informative and concise state representations. In spatial navigation, SRL can be employed through initial task exploration (MERCKLING et al., 2022).

A central RL algorithm is the Successor Representation (SR) vastly employed in the predictive map theory (DAYAN, 1993; RUSSEK et al., 2017; DUCAROUGE; SIGAUD, 2017; de Cothi et al., 2022). SR enables the independent learning of a predictive map of places (or states) and the potential reward within the environment. In the SR algorithm, RL states play a vital role as they serve as the foundation for representing the environment and capturing its temporal dynamics. By observing state transitions, the algorithm learns how the environment evolves and how the agent's actions influence these transitions. Notably, the original report of SR demonstrated its capacity to handle latent learning simulations (DAYAN, 1993).

Recently, Diekmann et al. (2023) have developed the closed-loop simulator for complex behavior and learning based on RL and deep neural networks (CoBeL-RL). The CoBeL-RL includes crucial mechanisms for neuroscientific plausibilities, such as replay and complex 3D environments. The framework is a prominent research tool, providing a range of RL agents, including an SR implementation and tools to monitor and analyze agent behavior. A notable feature of the framework is its flexibility, allowing modification of its modules to incorporate new functionalities. CoBeL-RL has demonstrated a latent learning effect in environments resembling those studied by Blodgett (1929).

Incorporating SRL methods can enhance CoBeL-RL. In spatial navigation, SLAM solutions such as RatSLAM can handle high-dimensional information such as images and velocities and transform them into spatially informative states such as position and orientation while exploring unknown environments. This dimensionality reduction can increase the efficiency of policy learning. By incorporating these capabilities, CoBeL-RL gains the ability to examine new paradigms that balance autonomous exploration and learning, such as artificial curiosity (PATHAK et al., 2017).

1.4 Objectives and Contributions

This work proposes a suite of biologically inspired approaches to building spatial maps for spatial navigation and learning. It also allows for building theoretical and empirical bases that can be explored for robotics and behavioral neuroscience applications. Specifically, we contribute to the following:

- Inspired by the biological mechanisms presented in (CARPENTER et al., 2015; WERNLE et al., 2018), we expand the RatSLAM in achieving mapping across multiple sessions, allowing virtual or real robots to build maps incrementally with RatSLAM (MENEZES et al., 2023);
- A novel approach to automatically tuning RatSLAM parameters by optimization processes (MENEZES et al., 2020; GOMES et al., 2022).
- Extend the capabilities of CoBeL-RL (DIEKMANN et al., 2023) by integrating RatSLAM as an SRL to enable automatic mapping for unknown environments;
- Building on the proposed combination of CoBeL-RL and RatSLAM, we model latent learning experiments and investigate how distinct exploration approaches during latent learning accelerate the learning of reward locations, as observed in rats. (KARN; JR., 1946).

Combining CoBeL-RL's learning mechanisms with RatSLAM's mapping abilities has significant implications for biological studies, particularly spatial navigation. This integration enables the design of realistic simulations that accurately model the learning processes involved in unfamiliar environments. Additionally, it can facilitate operating state-dependent RL algorithms such as the SR. By leveraging pre-exposure designs in latent learning, this research contributes to a deeper understanding of how agents navigate effectively in space. Overall, this work advances the fields of computational neuroscience and robotics by showcasing the potential of biologically-inspired approaches in map-building and learning.

This thesis is structured into the following chapters: Chapter 2 delves into the theoretical foundations needed for this study. Chapter 3 provides an overview of the relevant works related to this thesis. In Chapter 4, the methodology and results of RatSLAM's extension to handle multisession mapping are presented. Chapter 5 introduces the framework that combines RatSLAM and the CoBeL-RL for mapping and learning in unknown environments. It also investigates different pre-exposure designs and exploration strategies in latent learning. Finally, Chapter 6 presents the conclusions drawn from this thesis.

2 Theoretical Foundation

This Chapter provides an overview of the theoretical foundations supporting this work. Firstly, we provide an overview of Simultaneous Localization and Mapping (SLAM) in mobile robotics. Next, we introduce the RatSLAM algorithm, discussing the key concepts and principles. Thereafter, we delve into Reinforcement Learning and explore related algorithms such as Model-free, Model-based, and Successor Representation. Lastly, we discuss the CoBeL-RL, a closed-loop simulator of complex behavior and learning, highlighting the main components and overall functionality.

2.1 SLAM

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics and artificial intelligence that addresses the challenge of a robot or a mobile platform simultaneously mapping an unknown environment while estimating its position within that map (DURRANT-WHYTE; BAILEY, 2006; THRUN, 2008). To address this problem robustly, SLAM involves utilizing sensor measurements from various sources, such as odometry, laser range finders, or cameras (ZAFFAR et al., 2018). This process entails solving a sophisticated optimization problem incorporating these sensor measurements, motion models, and probabilistic estimation techniques (THRUN, 2008). Moreover, SLAM algorithms typically employ a combination of filtering, smoothing, or optimization methods to iteratively refine both the map and the robot's pose estimate, ensuring accuracy and consistency (MONTEMERLO et al., 2003; DURRANT-WHYTE; BAILEY, 2006; GRISETTI; STACHNISS; BURGARD, 2007).

The SLAM can be formally described in probabilistic terms (THRUN, 2008). When a robot is on a ground surface, its position at time t is represented by two-dimensional coordinates in the plane x_t and an orientation value. The set of coordinates is defined as:

$$X_T = x_0, x_1, x_2, \dots, x_T \tag{2.1}$$

where T represents a terminal time, and the initial location x_0 is unknown.

The odometry data, u_t, provides information about the robot's motion between time $t-1$ and t. This data sequence is represented as:

$$U_T = u_0, u_1, u_2, \dots, u_T. \tag{2.2}$$

Due to the accumulation of noisy measurements, odometry information becomes increasingly inaccurate over time. As a result, relying solely on odometry data is inadequate for accurately reconstructing the past trajectory X_T from the initial location x_0.

The environment map represented as $m = m_1, m_2, \ldots, m_N$ is assumed to be time-invariant, modeling a static environment. If we assume the robot takes measurements of features in m at each time point, the relationship between these measurements and the robot's pose x_T can be expressed as:

$$Z_T = z_0, z_1, z_2, \ldots, z_T \tag{2.3}$$

From the previous definitions, SLAM aims to recover m and X_T from the odometry data and environment measurements. Two primary forms of the SLAM problem are distinguished: the *full* SLAM problem and the *online* SLAM problem. The *full* SLAM problem involves calculating the probability of the robot's pose throughout the entire path, including the map.

$$p(X_T, m|Z_T, U_T) \tag{2.4}$$

On the other hand, the *online* SLAM algorithms recover the current robot location rather than the entire path. The online form is defined as:

$$p(x_t, m|Z_T, U_T) \tag{2.5}$$

SLAM algorithms for online data processing are usually incremental and analyze one data flow at a time (THRUN, 2008). Moreover, full and online SLAM algorithms employ filtering or optimization techniques to update the distribution as new data becomes available continuously.

2.2 RatSLAM

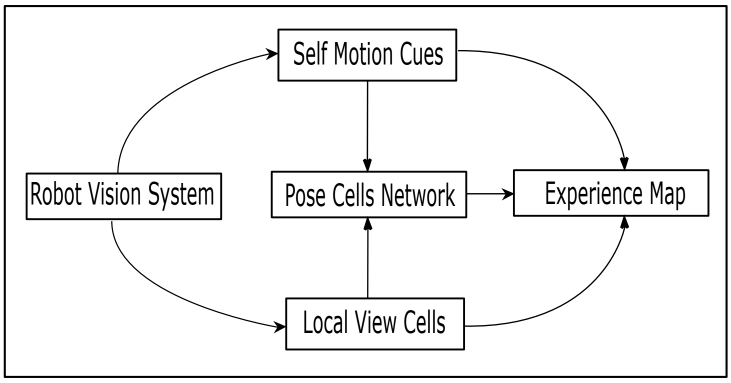
Milford and collaborators developed RatSLAM in 2004 as a localization and mapping solution for mobile robots, utilizing a vision system as the primary sensor input (MILFORD; WYETH; PRASSER, 2004). An updated architecture of RatSLAM, depicted in Figure 2, has been proposed by Ball et al. (BALL et al., 2013). The architecture consists of five modules to be detailed subsequently, together with the influence of RatSLAM parameters on each module.

- The **Robot Vision System** is responsible for capturing images and transmitting them to other modules.
- The **Self Motion Cues** module estimates translational and angular velocities based

on robot odometry. This information is then forwarded to the Pose Cell Network and Experience Map. Additionally, visual odometry can be computed using the images from the Robot Vision System (Fig. 2, dashed line).

- The **Pose Cells Network** consists of a three-dimensional Continuous Attractor Network (CAN) comprising units connected by excitatory and inhibitory connections. Each cell within the Pose Cell Network represents the robot's position (x, y) and orientation θ.
- The **Local View Cells** maintain a list of view scenes, referred to as templates. A new template is created based on whether the scene received from the Robot Vision System is novel. An association between the template and the activity in the Pose Cells Network is learned upon template creation.
- The **Experience Map** is a structured graph with Cartesian properties that serves as a topological-metric map representation of the environment.

Figure 2 – RatSLAM architecture.



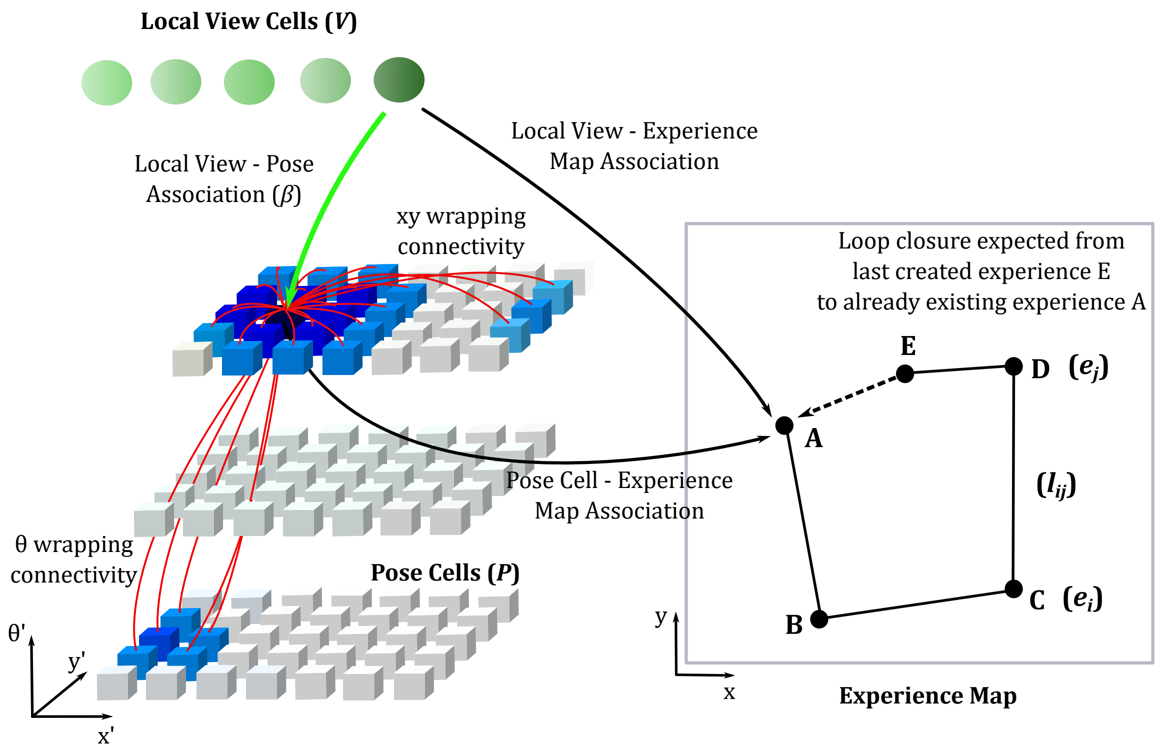
Source: Designed by the author and adapted from (MENEZES et al., 2023)

2.2.1 Pose Cells Network

The Pose Cells Network (PCN), denoted as P, is a three-dimensional continuous attractor network (CAN) that represents the robot's position (x', y') and orientation θ in 2D space (Fig. 3). The PCN units link through excitatory and inhibitory connections, extending across all six PCN faces (red arrows in Fig. 3). This feature enables the network to operate beyond its fixed size limitations, virtually representing environments larger than the PCN can directly encode. Furthermore, each cell within the PCN has an associated value that reflects its activity on the network.

Local excitatory connections enhance the activity of neighboring cells within the PCN. On the other hand, global inhibitory connections suppress the activity of distant cells

Figure 3 – Interconnections among the components of RatSLAM.


Local View Cells (V)
Local View - Pose Association (β)
Local View - Experience Map Association
xy wrapping connectivity
Loop closure expected from last created experience E to already existing experience A
E
D (e_j)
A
(l_{ij})
Pose Cell - Experience Map Association
θ wrapping connectivity
Pose Cells (P)
C (e_i)
B
θ'
y'
x'
y
x
Experience Map

Source: Designed by the author and adapted from (MENEZES et al., 2023)

or those that exhibit low levels of activity within the network. This dynamic is described by the distribution ε (MILFORD; WILES; WYETH, 2010):

$$\varepsilon_{a,b,c} = e^{-(a^2+b^2)/k_p^{exc}} e^{-c^2/k_d^{exc}} - e^{-(a^2+b^2)/k_p^{inh}} e^{-c^2/k_d^{inh}} \tag{2.6}$$

where k_p and k_d are the variance constants for place and direction, respectively. Additionally, the parameters a, b, and c represent the distances between the coordinates of two cells, taking into account the periodic boundary conditions of the network. The distance between a cell with coordinates x', y', and θ and another cell with coordinates i, j, k can be calculated as follows (BALL et al., 2013):

$$\begin{aligned} a &= (x' - i)(\text{mod } n_{x'}), \\ b &= (y' - j)(\text{mod } n_{y'}), \\ c &= (\theta' - k)(\text{mod } n_{\theta'}), \end{aligned} \tag{2.7}$$

where “mod” represents the modulo operator. The parameters $n_{x'}$, $n_{y'}$, and $n_{\theta'}$ indicate the size of the network in terms of the number of cells along the X', Y', and Θ' dimensions respectively. The change of activity in a cell is given by:

$$\Delta P_{x',y',\theta'} = \sum_{i=0}^{n_{x'}-1} \sum_{j=0}^{n_{y'}-1} \sum_{k=0}^{n_{\theta'}-1} P_{i,j,k}\varepsilon_{a,b,c} - \varphi \tag{2.8}$$

where φ is the global inhibition, the final step in the network update involves constraining the activation levels in P to non-negative values and ensuring that the total activation is normalized to one (MILFORD; WILES; WYETH, 2010).

Over time, PCN dynamic facilitates the formation of clusters of activated cells within the CAN, commonly referred to as "energy packets" or "activity packets" (Fig. 3, blue cubes) (BALL et al., 2013). The center of the energy packet represents the closer estimation of the PCN to the robot's pose within the environment (Fig. 3, darker blue cube). Furthermore, the energy packet's direction changes based on odometry information. Additionally, templates stored in LVC can inject energy into the PCN and cause the activity packet to "jump" to a different location.

2.2.2 Local View Cells

The Local View Cells (LVC), V, build an array of *templates*, denoted by V_i. Each template aims to capture a unique environment representation from the Robot Vision System data. Moreover, when creating a template, a short learning excitatory connection β establishes a link between it and the center of the dominant activity packet in the PCN (BALL et al., 2013). Consequently, β links the robot's estimated pose in PCN to the distinct view from that location. The link is given by (MILFORD; WILES; WYETH, 2010):

$$\beta^{t+1}_{i,x',y',\theta'} = \max(\beta^{t}_{i,x',y',\theta'}, \lambda V_i P_{x',y',\theta'}) \tag{2.9}$$

where lambda is the learning rate, and i refers to the activated template V_i. Moreover, x', y', and θ' indicate the PCN's dominant energy package coordinate. Note that $\beta^{t+1}_{i,x',y',\theta'} = \lambda V_i P_{x',y',\theta'}$ only if there is no prior association between the template and energy package.

When the RatSLAM detects the robot has returned to a previously visited location, it reactivates stored templates. This process, known as loop closure, injects energy into the PCN at specific coordinates (x', y', θ') associated with the learned link β and the activated template V_i:

$$\Delta P_{x',y',\theta'} = \delta \sum_{i} \beta_{i,x',y',\theta'} V_i \tag{2.10}$$

where δ is the constant that determines the influence of visual features on the estimated robot's pose. When the PCN receives a constant injection of activity, its dominant energy package shifts, resulting in a corresponding change in the representation of the robot's position and orientation within the network (BALL et al., 2013).

2.2.3 Experience Map

The Experience Map (EM) is a structured graph that combines the PCN activity and LVC templates to estimate robot poses in a two-dimensional map. A *experience* node in the EM is defined as a 3-tuple (BALL et al., 2013):

$$e_i = \{P_i, V_i, \mathbf{p}_i\} \tag{2.11}$$

where $\mathbf{p}_i$ is the robot's pose in the experience map associated with P_i and V_i states, respectively. The RatSLAM builds a new experience when both P_i and V_i do not closely match with states of any existing experience. Otherwise, the experience is reactivated.

In addition, a link $l_{i,j}$ stores distance and temporal information between the transition from experience. The temporal information might be suitable for path planning from a specific experience to the desired goal. For example, Dijkstra's algorithm can be used to find the shortest path between two nodes (BALL et al., 2013). A link built from experiences e_i to e_j is defined as:

$$l_{i,j} = \{\Delta\mathbf{p}_{i,j}, \Delta t_{i,j}\} \tag{2.12}$$

where $\Delta\mathbf{p}_{i,j}$ and $\Delta t_{i,j}$ are the distance and time interval between the two experiences, respectively.

The EM processes described above rely mainly on the robot's odometry information. However, when loop closure is detected, the robot re-localizes within the map, reactivating a prior experience. To ensure accuracy on the experiences' pose, an algorithm iteratively corrects odometry errors based on the reactivated node's position. This leads to adjustments in the poses of past experiences, which are calculated as follows:

$$\Delta\mathbf{p}_i = \alpha \left(\sum_{j=1}^{N_f} (\mathbf{p}_j - \mathbf{p}_i - \Delta\mathbf{p}_{i,j}) + \sum_{k=1}^{N_t} (\mathbf{p}_k - \mathbf{p}_i - \Delta\mathbf{p}_{k,i}) \right) \tag{2.13}$$

where α is a correction rate constant set to 0.5, N_f is the number of links from experience e_i to other experiences, and N_t is the number of links from the different experiences to the experience e_i.

2.2.4 Role of RatSLAM Parameters

The RatSLAM algorithm has various parameters that affect its mapping performance across all its modules (Tab. 1). These parameters control template operations, including comparison, size dimensions, and conditions for creating new templates in the Local View module. The Pose Cells module parameters determine the dynamics and dimensions of the network, such as values for local excitation, global inhibition, and energy injection during loop closure. The Experience Map parameters define the number of iterations for the graph correction algorithm.

Consequently, incorrectly adjusted parameters can cause RatSLAM to malfunction, resulting, for example, in false loop closure detection, undesirable creation of new experiments and templates, and excessive delay in algorithm execution.

Moreover, the Self Motion Cues module (Fig. 2) provides odometry information obtained from the Robot Vision System. The parameters must be adjusted for visual odometry to accurately estimate the robot's translational and rotational velocities. However, the setting of these parameters can be omitted if alternative sources, such as wheel encoders, provide the odometry information.

2.3 Reinforcement learning

Reinforcement Learning (RL) is a widely recognized learning paradigm that involves an agent learning how to maximize a numerical reward signal value while interacting with an environment (SUTTON; BARTO, 2018). Unlike other learning approaches, RL does not depend on explicit instructions for the agent's actions during the interaction. Instead, it learns through experiences, engaging in a trial-and-error process to determine the most effective choice. This type of learning is related to what has been observed in the neuroscience field (SUBRAMANIAN; CHITLANGIA; BATHS, 2022). The concept of RL draws inspiration from understanding how organisms learn from their environment and adjust their behavior accordingly. Neuroscientific studies have revealed reward-related brain mechanisms that influence decision-making and learning processes, which aligns with the principles of RL (SCHULTZ, 2000).

2.3.1 Elements of Reinforcement Learning

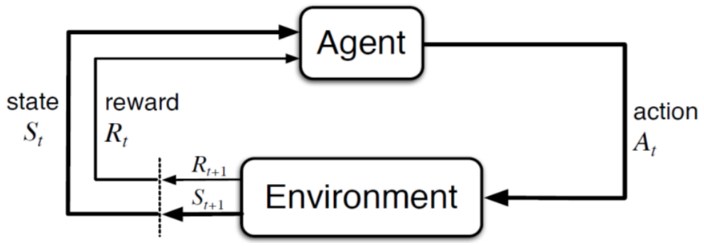
The RL setup holds formalized elements (SUTTON; BARTO, 2018). First, the learner in this setup is called the *agent*, which interacts with the *environment* over time. As the agent interacts, it observes a series of *states* that represent the current conditions of the environment. Then, the agent can choose and conduct a particular *action* using the information from the current state. The agent's action may induce changes in the environment, and in response, the agent receives a *reward* as evaluative feedback for

Table 1 – RatSLAM Parameters

Module	Parameter's Name	Type
Visual Odometry	[vtrans_image_x_min, vtrans_image_x_max, vtrans_image_y_min, vtrans_image_y_max];	Integer
	[vrot_image_x_min, vrot_image_x_max, vrot_image_y_min, vrot_image_y_max];	
	camera_fov_deg;	
	camera_hz;	
	vtrans_scaling;	Real
	vtrans_max.	
Local View Cells	vt_panoramic	Binary
	vt_shift_match;	Integer
	vt_step_match	
	[image_crop_x_min image_crop_x_max image_crop_y_min image_crop_y_max]	
	[template_x_size, template_y_size]	
	vt_match_threshold;	Real
	vt_normalization;	
	vt_patch_normalization.	Integer
Pose Cells Network	pc_dim_xy	
	exp_delta_pc_threshold	Real
	pc_cell_x_size	
	pc_vt_inject_energy	
	vt_active_decay	
	pc_vt_restore	Integer
Experience Map	exp_loops	

its chosen behavior. Typically, a positive reward signal is linked to receiving positive reinforcement, while a negative call is associated with obtaining a punishment. To model and analyze the agent-environment interaction, the Markov Decision Process (MDP) framework is commonly employed (Fig. 4).

Figure 4 – Agent-environment interaction in a Markov decision process.



Source: Adapted from (SUTTON; BARTO, 2018)

In RL, the agent's actions have consequences beyond immediate rewards. They can influence the subsequent states of the environment, leading to a chain of connected situations and impacting all future rewards. This sequential nature of RL allows the agent to develop a long-term strategy by considering the future implications of its decisions.

To maximize the overall reward signal value, the agent prioritizes the accumulation of rewards throughout an *episode*, which refers to the duration between the initiation of the learning process and the achievement of a possible final state. A challenge arises when the rewards in an episode are sparse, i.e., it will be presented only in the uncertain future, making it difficult for the agent to acquire information about distant rewards in the long run. For example, imagine an agent deciding between two actions while navigating a maze. If it chooses to turn to the right, the agent immediately receives a small amount of positive value as a reward. On the other hand, if it turns left, there is no immediate reward. Nevertheless, after a few more actions, the final reward prize is considerably more significant than the initial one.

RL agents can use value functions to address this challenge, which estimates the total future reward an agent can expect to receive given a particular policy starting from a specific state. The policy is a stochastic rule that guides the agent's action selection based on observed states, driving its behavior within the environment and specifying the probability distribution of actions in different states.

Importantly, RL policies must deal with the exploration-exploitation dilemma, which involves finding the optimal trade-off between exploring new actions and exploiting the knowledge gained from previous actions. Examples of policies widely adopted in RL are ε-greedy and *softmax*

The ε-greedy, policy is a simple yet effective approach where the agent selects the action with the highest estimated value (greedy action) most of the time. However, the agent chooses a random action with a ε (epsilon) probability to encourage exploration. This randomness allows the agent to explore different actions and find more valuable strategies. Conversely, the softmax policy, known as the Boltzmann exploration (SUTTON; BARTO, 2018), assigns probabilities to each action based on their estimated values. The probabilities are computed using the softmax function, which gives higher probabilities to actions with higher values. However, unlike epsilon-greedy, softmax policy still assigns non-zero probabilities to suboptimal actions, allowing the agent to explore them to some extent.

Both policies aim to find a balance between exploration and exploitation. Epsilon-greedy has a higher chance of exploiting the current best action while occasionally exploring random actions. *Softmax*, on the other hand, explores actions proportionally to their estimated values, providing a softer exploration strategy. Ultimately, RL aims to find an optimal policy, allowing the agent to make the best decisions in each state by maximizing the expected long-term reward. We formalize the value function of a state $V(s)$ under the policy π as follows:

$$V_\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_t | s_0 = s] \tag{2.14}$$

where $\mathbb{E}_\pi$ represents the expected value of a random variable when the agent follows a specific policy π, and the variable t is any time step. The $\gamma \in [0, 1]$ is the discount factor, impacting the ratio between immediate and future rewards in the agent's decision-making process. When $\gamma = 0$, the agent prioritizes immediate rewards. On the other hand, when $\gamma = 1$, future rewards become more significant, motivating the agent to maximize cumulative rewards over time and prioritize actions that yield long-term advantages. The value for the discount factor depends on the specific problem. A lower γ is suitable for tasks where immediate rewards are critical, such as environments with rapid changes. On the other hand, a higher γ is appropriate for tasks involving long-term planning and significant implications of delayed rewards.

In addition to value functions, the RL framework extends this concept to *action-value functions*, also known as Q-values and denoted as $Q_\pi(s, a)$, which incorporate the actions taken by the decision-making agent. Q-values estimate the expected cumulative reward that the agent can achieve by taking a specific action a from state s and subsequently following a particular policy π. Unlike value functions, which focus only on states, Q-values measure the long-term potential and immediate rewards of choosing a particular action in a given state:

$$Q_\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_t | s_0 = s, a_0 = a] \tag{2.15}$$

2.3.2 Model-free and Model-based RL

In RL, two fundamental approaches can be employed for an agent to learn a given task. The first, model-free RL, involves learning the optimal policy directly through a trial-and-error scheme, updating the value or action-value functions at each new experience. Alternatively, RL can use models of the environment to predict the outcomes of its actions and select the action that yields the best result. By leveraging an internal model, the agent can plan over potential paths. In the second approach, known as model-based RL, actions are chosen based on predictions made by a model. This model anticipates the outcomes of different actions regarding future states and the corresponding expected rewards (SUTTON; BARTO, 2018).

Both approaches have their advantages and trade-offs, with model-free algorithms offering fast learning but limited ability to plan in case of changes in the environment and model-based RL providing planning capabilities but requiring an accurate model of the environment.

Model-free RL

In model-free RL, the agent learns to make decisions and improve performance through trial-and-error experiences. This method focuses on learning without building an explicit model of the environment. Instead, the agent directly interacts with the environment, obtains rewards for its actions, and adjusts its policy accordingly. Model-free agents depend on value functions to evaluate and update the current policy over new experiences since they measure the value of expected future rewards for the following states and guide the agent's decision-making process. The Bellman equation states the decomposition of a current state value $V(s_t)$ in terms of the expected reward (r_t) and the value associated with the next state, as can be seen in:

$$\begin{aligned} V(s) &= \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3}] \quad (2.16) \\ &= \mathbb{E}[r_t + \gamma V(s_{t+1})] \quad (2.17) \end{aligned}$$

One critical component of model-free RL is the capability to update the value function as the agent gains additional experience. The value function is iteratively adjusted using the temporal-difference (TD) learning rule. This rule involves comparing the predicted outcomes of actions with the actual observed results and updating the value estimates accordingly (SUTTON; BARTO, 2018):

$$V(s) \leftarrow V(s) + \alpha[r_t + \gamma V(s_{t+1}) - V(s)] \quad (2.18)$$

where $V(s)$ represents the value estimate for state s, r_t is the reward received at time step t, α is the learning rate, and γ is the discount factor that balances the importance of immediate and future rewards.

Furthermore, model-free RL can employ on-policy and off-policy methods for learning and updating the agent's policy. On-policy algorithms involve learning the value function or policy based on the data collected by the agent while following its current policy. The updates are directly based on the actions taken by the agent during its exploration. In contrast, off-policy methods enable the agent to learn and update its value functions and policy using data generated by a different policy. This means that the agent can leverage experiences collected under a different exploration policy to improve learning. One notable example of an off-policy algorithm is Q-learning, which updates the Q-values by estimating the maximum expected future rewards for each state-action pair. The update rule in Q-learning involves selecting the maximum Q-value ($\text{max}_a Q(s_{t+1})$) for the next state, regardless of the action selected by the policy and taken by the agent (Eq. 2.19).

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r_t + \gamma \text{max}_{a'} Q(s_{t+1}, a') - Q(s,a)] \tag{2.19}$$

In large state or action spaces, function approximation techniques are often used to approximate the value function or Q-values (SUTTON; BARTO, 2018). Deep Q-Network (DQN) is a well-known model-free RL algorithm that combines Q-learning with DNN (MNIH et al., 2015). It leverages a neural network to approximate the Q-values. It employs an experience replay mechanism, where past experiences. During learning, batches of experiences are randomly sampled from the replay buffer. Furthermore, DQN employs a form of epsilon-greedy exploration, i.e., the agent explores the environment by randomly selecting actions. As the agent's training progresses, it gradually shifts towards exploiting its knowledge by selecting actions with the highest estimated Q-values.

Model-based RL

Model-based RL involves learning a model that captures the dynamics of the environment or the underlying process controlling the agent's interactions. This model enables the agent to simulate different outcomes resulting from its actions, facilitating selecting the action that maximizes the expected return. This approach is particularly effective in simple and predictable environments. However, as the state/action space expands, predicting each outcome becomes increasingly challenging and computationally intractable.

The model employed in model-based RL can be stochastically designed to address the inherent randomness and uncertainty of the real scenarios. In this scenario, the environment may respond to actions in numerous ways, each with its probability of occurrence and allowing the agent to make robust and adaptable decisions to the dynamic nature of the environment. Moreover, the learned model's accuracy significantly impacts the model-based RL's effectiveness. If the model inaccurately represents the actual dynamics of the environment, it can lead to suboptimal decision-making by the agent.

One of the simplest yet general ways to represent a valuable model of the environment is through the state transition probabilities. Mathematically, the state transition probability, denoted as $P(s'|s,a)$, represents the likelihood of transitioning from state s to state s' when taking action a. It can be formulated as follows:

$$P(s'|s,a) = Pr[s_{t+1} = s'|s_t = s, a_t = a], \tag{2.20}$$

where s_t and a_t are the state and action at time t, respectively. Alternatively, a one-step transition matrix can define the environment model through state transition probabilities. This representation, $T(s, s') = Pr(s_{t+1} = s'|s_t = s)$, is a square matrix with each row representing a state s and containing the probabilities of transitioning to state s' in a single step. The sum of probabilities in each row, $\sum_{s'} T(s, s')$, adds up to 1, ensuring a valid transition model.

The one-step transition matrix can be used for *sample* model-based planning procedures. This approach involves generating single or multiple trajectories by iteratively sampling actions according to the policy dictated by T and observing the resulting states. The sample model-based planning usually involves iteratively updating the estimates based on new samples and refining the policy accordingly. This iterative improvement allows the agent to learn from simulated experiences and adjust its decision-making strategy for better environmental performance.

Another approach is to approximate the value function directly using T to avoid the excessive computational overhead associated with model-based planning. The Successor Representation method uses T to build a predictive map of the environment.

2.3.3 The Successor Representation

The Successor Representation (SR) algorithm, introduced by Dayan (1993), captures the expected future visits of states to facilitate learning. It offers an alternative perspective on modeling the environment dynamics compared to traditional methods focusing on explicit transition models, such as model-based algorithms. Rather than explicitly modeling state transitions, the SR algorithm aims to understand the underlying dynamics of the environment by storing information about the expected future visitation of states from the current state.

The SR algorithm builds a matrix, M, to track estimations. Given a known state transitions matrix of the environment T, M is computed as a discounted sum over T raised to the time step t:

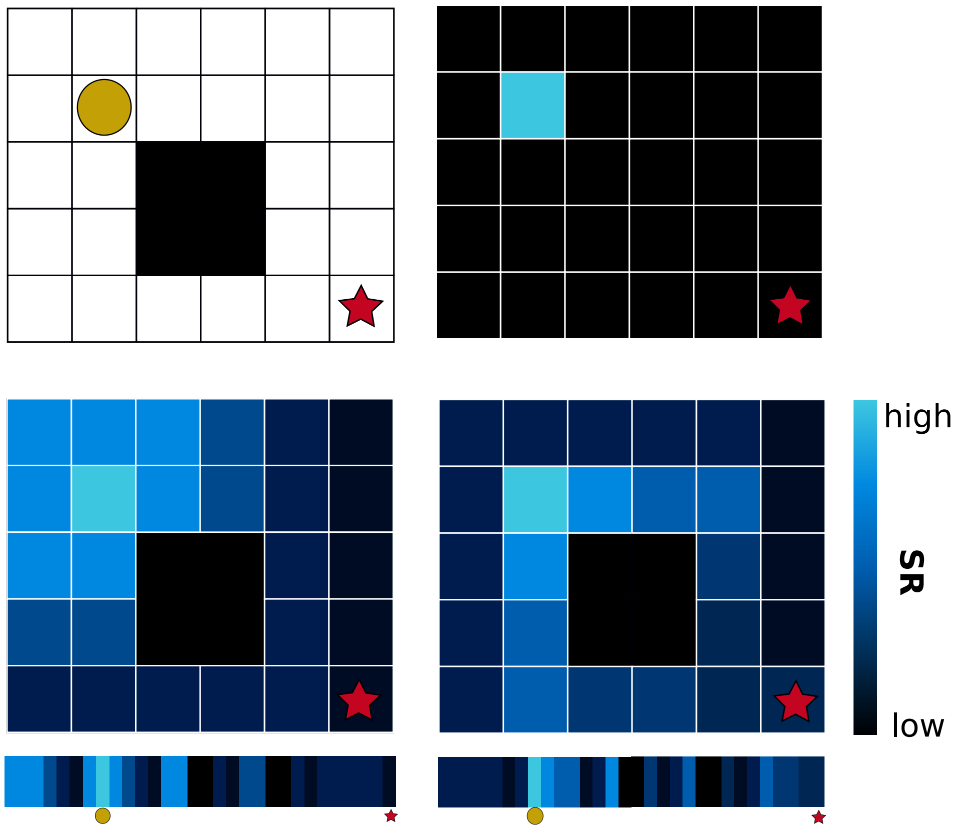
$$M = I + \gamma T + \gamma^2 T^2 + \gamma^3 T^3 ... \tag{2.21}$$

$$= \sum_{t=0}^{\infty} \gamma^t T^t \tag{2.22}$$

where γ dictates the temporal discounting of future state occupancy. Each row of the matrix represents the expected visitation frequencies of all states given the current state. By updating this matrix during the learning process, the SR agent accumulates knowledge about the likely future states it will encounter.

To illustrate the concept, let us consider a grid world scenario with an SR agent represented as a circle and a reward represented as a star (Fig. 5). The blue light in the top-right frame indicates the agent's current state in the 2D space. Based on its current policy, the agent can gain insights into the expected discounted visits to other states using the SR algorithm. Therefore, the occupancy predictions for future states will vary depending on the chosen policy. For a randomly uniform exploration policy, the occupancy predictions for the current state will resemble the lower-left frame. However, if the agent follows a policy that maximizes the reward, the occupancy prediction of future states will align with the lower-right frame. Single arrays below each matrix represent the state row in M.

Figure 5 – SR occupancy grid.



Source: Designed by the author

The value function for a state $V(s)$ can be define as the multiplication of the SR matrix $M(s, s')$ with the environment's reward function $R(s)$ (STACHENFELD; BOTVINICK; GERSHMAN, 2017):

$$V(s) = \sum_{s'} M(s, s')R(s'), \tag{2.23}$$

where $M(s, s')$ encodes the expected discounted future occupancy of state s' along a trajectory initiated in state s:

$$M(s,s') = \mathbb{E}[\sum_{t=0}^{\infty}\gamma^t \mathbb{I}[s_t = s']|s_0 = s] \tag{2.24}$$

$\mathbb{I}[.] = 1$ if s' is reachable from s_t and 0 otherwise.

The SR is independent of the reward function, enabling rapid adaptation to changes in the environment's reward structure. Thus, the agent only needs to relearn the new reward function $R(s)$ while maintaining the existing SR M, resulting in fast integration of updated reward signals. By separating the representation of the environment from the reward function, the SR facilitates efficient learning and decision-making in dynamic environments. Moreover, as the agent acquires new experiences over interactions, M and $R(s)$ are updated with temporal-difference learning rules. Consider the agent goes from $s_t \rightarrow s_{t+1}$ and receives reward r, then the agent can implement the learning rules:

$$M(s_t,s') \leftarrow M(s_t,s') + \alpha[\mathbb{I}s + \gamma M(s_{t+1},s') - M(s_t,s')] \tag{2.25}$$

$$R(s') \leftarrow R(s') + \alpha[r - R(s')] \tag{2.26}$$

where α is a defined learning rate.

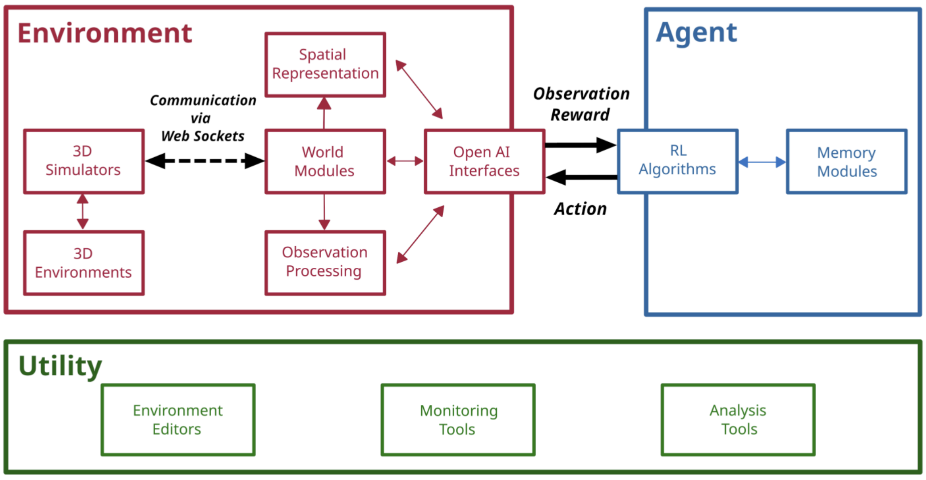
Furthermore, Deep Successor Reinforcement Learning (DSR) (KULKARNI et al., 2016) offers a function approximation method for learning the successor representation algorithm. To do so, DSR enables the estimation of the successor representation (SR) and reward function by DNN with raw sensory observations as input. DSR computes the inner product between the SR and predictions of immediate rewards to estimate the value function.

2.4 CoBeL-RL

The “Closed-loop simulator of Complex Behavior and Learning based on Reinforcement Learning and deep neural networks”, or CoBeL-RL for short, is a neuroscience-oriented framework for efficiently setting up the closed-loop interaction between an agent and an environment (Fig. 6). It focuses on trial-based experimental designs, and each trial is further differentiated into agent-environment interactions referred to as steps. Each step yields an experience tuple (s, a, r, s'), which the agent can learn directly from or store in a memory structure for later learning.

CoBeL-RL has been structured to integrate modules that allow large flexibility in building different learning setups. These modules are *Environment*, *Agent*, and *Utility*.

Figure 6 – The CoBeL-RL architecture.



Source: Adapted from (DIEKMANN et al., 2023)

2.4.1 Environment Module

CoBeL-RL offers two types of environments: simple implementations, such as grid worlds, and complex environments rendered by popular game engines like Blender Game Engine (Blender Online Community, 2018), Godot (LINIETSKY; MANZUR, 2007), or Unity (Unity Technologies, 2005). CoBeL-RL also includes additional modules such as the 3D simulator module, an observation module for preprocessing observations, and *spatial representation* modules for navigation to facilitate interaction with the game engines.

The CoBeL-RL's *spatial representation module* enhances agents' navigation capabilities within the environment by using a simplified spatial representation rather than relying on continuous physical movement. Currently, it creates a topological graph of the environment, where nodes represent distinct locations and edges establish their connectivity. Users can manually define the topological graph using the Blender Game Engine (BGE) or automatically generate it using the grid graph module.

Furthermore, the spatial representation module also defines the agent's action space and specifies how actions correspond to transitions on the graph. The default topological graphs in CoBeL-RL support two transition modes. The first mode, "without rotation," limits transitions to neighboring nodes exclusively. In this mode, agents can move between adjacent nodes without rotational movement. The second mode, referred to as “with rotations”, permits both translational and rotational movements on the graph.

The *observation module* in CoBeL-RL provides functionality for preprocessing environment observations before transmitting them to the RL agent. This module retrieves the agent's x-y position coordinates and current heading direction in its simplest form. The RL agent can utilize these values independently or with visual observations. Additionally, the observation module preprocesses visual observations, including resizing them to a user-defined size and normalizing pixel values within the [0, 1] range, enabling efficient transmission to the agent.

The observation module can additionally introduce various types of noise, such as Gaussian noise, to capture the inherent imprecision found in biological observations. This feature enhances the realism of the learning environment, enabling the RL agent to adapt and learn from noisy sensory inputs. Furthermore, the observation module supports combining two or more observations, allowing for the simulation of multisensory observations.

The *3D Simulators module* bridges the CoBeL-RL framework and game engines to enable simulation and rendering. This module facilitates communication between the framework and game engines via web sockets. CoBeL-RL supports three game engines for simulation and rendering: BGE, Unity, and Godot.

For a particular interest in this study, we delved into the BGE Simulator. The default BGE Simulator module utilizes three web sockets for communication. The control socket handles the transmission of commands and retrieval of relevant control data, such as object identifiers. Commands are encoded as strings and follow a comma-separated format, containing the command name and corresponding parameter values. Likewise, retrieved values are received in string format and are separated by commas. Visual observations are obtained through the video socket, while sensory data is retrieved via the data socket. Finally, a new Blender process is launched during module initialization, and a user-defined Blender scene is opened.

2.4.2 Agent Modules

The RL agents are implemented via the Agent modules that define their behavior, including learning, exploration strategies, and memory. All agents inherit from a common abstract RL agent class and must implement functions for training, testing, and computing predictions for a given batch of observations (DIEKMANN et al., 2023). Callbacks are also implemented by agents to allow for fine-grained control during simulations. Information such as the number of trial steps and rewards are collected by the agents and passed to the callbacks. Custom callback functions can be defined by the user and passed as a dictionary to the RL agent.

RL agents in CoBeL-RL can use the experience replay (LIN, 1992; MNIH et al., 2015) as part of the learning algorithm. These agents are connected with different memory modules that operate as buffers for experience replay. Experience tuples are stored in the memory modules, providing the possibility of building up a history of experiences, which are used for training. Agents and memory modules can be freely combined to study the effects of different replay models.

Here, we mention four available agents in CoBeL-RL: DQN agents, Dyna-Q, and hybrid Dyna-DQN and DSR agents.

DQN agent

DQN agent encapsulates Keras-RL2's implementation. It uses a small, fully connected DNN by default and follows an epsilon-greedy policy. There are also versions of DQN that implement Prioritized Experience Replay (SCHAUL et al., 2016) (PER-DQN) and learn an environmental model.

Dyna-Q agents

The Dyna-Q model (SUTTON; BARTO, 2018) is implemented as a tabular agent. In this approach, the agent's Q-function is represented as an array with a size of $|S| \times |A|$, where $|S|$ denotes the number of environmental states and $|A|$ represents the number of available actions. Since this agent relies on a tabular representation, it can only be used in discrete static environments where states are represented as abstract indices, such as the grid world interface. The memory module is responsible for storing and retrieving experiences. Users can choose between an epsilon-greedy policy (the default) or a softmax policy for action selection.

Additionally, an optional action mask can exclude actions that do not result in a state change from the selection process. Moreover, the agent's Q-function is updated online with each step and can be updated via experience replay. Experience replay can be performed after each step, after each trial, or can be disabled altogether.

Dyna-DQN and DSR agents

CoBeL-RL introduces hybrid agents, specifically Dyna-DQN, which combine elements from the Dyna-Q agent and DQN. The DQN agent does not rely on the Keras-RL2. Instead, it can be implemented separately using TensorFlow 2 or PyTorch. The agent can receive a set of observations corresponding to discrete environmental states. If no observations are defined, a one-hot encoding of the environmental states is generated as the observations.

Furthermore, CoBeL-RL provides a hybrid agent called Dyna-DSR, which implements a variation of Deep Successor Reinforcement Learning (DSR) (KULKARNI et al., 2016) and bases our RL agents in the next chapters. However, unlike the DSR, the Dyna-DSR does not learn a separate feature representation of its observations. Instead, reward and SR models are trained directly on the observations. The resulting learned SR is called the deep successor representation (Deep SR).

2.4.3 Analysis Modules

The CoBeL-RL's Utility proved useful modules for monitoring, environment editing, and analysis of simulation variables, e.g., behavior and learning progress. These tools offer the capability to display and store important metrics in RL, such as the escape latency (the number of steps required to complete a trial), cumulative reward attained in each trial, responses emitted by the agent in each trial or step, and the trajectory of the agent. Additionally, for Deep RL agents, the response monitor can track the activity of units within the layers.

Furthermore, the environment editing feature allows users to create grid world environments by manually defining relevant variables, such as size, starting states, and reward functions. Alternatively, users can utilize templates for specific instances of grid world environments, such as an open field with a single goal location.

2.5 Summary

This chapter overviewed the fundamental concepts necessary to comprehend this work. We began by introducing the basics of SLAM and followed with a comprehensive explanation of RatSLAM and its structures. We delved into the Reinforcement Learning paradigm, discussing its key elements and introducing the distinction between model-free and model-based methods. Additionally, we presented the successor representation, which is the foundation for the agents employed in this study. Lastly, we introduced CoBeL-RL, on which this study's proposed framework is built.

3 Related Works

This chapter offers a literature review relevant to the doctoral thesis. We initially investigated studies on spatial map construction using neuro-inspired frameworks, with a specific focus on RatSLAM. Furthermore, the challenges RatSLAM faces in building maps in various SLAM sessions are discussed. Subsequently, we discuss how the brain may encode cognitive maps to facilitate adaptable behavior based on computational frameworks that show latent learning effects to facilitate adaptive behavior. Finally, an analysis is performed to assess the influence of various exploration design strategies on acquiring cognitive maps and their impact on learning speed in spatial navigation tasks.

3.1 Building Maps with Neuro-Inspired Frameworks

The mentioned neurological processes have inspired building neuro-inspired computational frameworks to solve real-world problems such as SLAM and understanding the underlying navigation processes in animals' brains through these models (YU et al., 2019; TANG; YAN; TAN, 2018; WANG; YAN; TANG, 2021; MILFORD; WILES; WYETH, 2010). In particular, we mention frameworks that build cognitive map models based on the integration of head direction cells and grid cells to perform path integration in 2D (MILFORD; WYETH; PRASSER, 2004; ZENG; SI, 2017; ZENG et al., 2020; WANG; YAN; TANG, 2021), or 3D SLAM (YU et al., 2019). Moreover, these algorithms rely on visual signals to adjust the robot's route when faced with inaccurate inputs when calculating path integration and performing loop closure. Ultimately, they combine the spatial information in a graph-like representation with a semi-metric relationship, which corresponds finds on the cognitive maps' underlying structure in the spatial context (PEER et al., 2021).

Our work focuses on RatSLAM, a well-known SLAM solution developed by Milford, Wyeth, and Prasser (MILFORD; WYETH; PRASSER, 2004; BALL et al., 2013) that incorporates visual sensing, robot movement, and pose cells, a 3D Continuous Attractor Network (CAN), to accurately estimate the robot's position and orientation during exploration. Continuous Attractor Network operates similarly to the conjunctive grid and head direction cells. RatSLAM selectively activates specific regions within the CAN by integrating visual information and odometry data. The algorithm builds an "Experience Map" combining visual input and CAN activity to represent the robot's path traveled in the environment.

3.2 The Multisession Problem for RatSLAM

RatSLAM has demonstrated successful performance in both indoor and outdoor SLAM tasks (PRASSER; MILFORD; WYETH, 2006; MILFORD; WYETH, 2008; MILFORD; WYETH, 2010; MENEZES et al., 2018; TANG; YAN; TAN, 2018; BALL et al., 2013). However, further development is still needed to enhance its capabilities in handling complex mapping scenarios, including creating a consistent global map across distinct environments (WERNLE et al., 2018). This challenge is closely associated with multisession SLAM, where robots conduct mapping across multiple sessions.

Multisession SLAM becomes crucial when mapping an entire environment in a single session becomes impractical, especially in expansive environments or when robots experience shutdowns followed by subsequent restarts. Moreover, multisession SLAM provides valuable solutions for addressing tracking failures that may occur in visual SLAM due to sensory occlusion.

In multisession SLAM approaches, the robot may initiate a new session from a random position in the environment and accurately incorporating this new position into the previously created map poses a significant challenge, commonly known as the “kidnapped robot” problem (MCDONALD et al., 2013; LABBÉ; MICHAUD, 2018). To address this problem, Labbé e Michaud (2018) propose two potential solutions: (i) the robot localizes itself on the existing map before starting the new session, or (ii) the robot begins mapping at the new location using its reference coordinates and subsequently merges this new map with the previously created maps to generate a unified representation. This work primarily focuses on the latter solution, investigating methods to effectively integrate the maps from different sessions into a cohesive and accurate representation.

Milford and Wyeth demonstrated using RatSLAM in multiple environments for autonomous mapping in their work (MILFORD; WYETH, 2010). Their robot performed delivery tasks for two weeks in different physical locations. It also can be moved between these locations without prior notification. When the agent moves to a new environment for the first time, it tries to locate itself on the previous map. If it fails to do so, RatSLAM creates a new Experience Map for the new location. The new map has the same RatSLAM structure but is topologically separate from the first map. This method keeps multiple local maps in the same RatSLAM structure, but if there is only one connection between them, the map can deform due to false metric representation. Therefore, this solution does not solve the multisession scenario stated in (ii).

In recent work, Tang, Yan e Tan (2018) developed a navigation system that uses the relationship between the hippocampus and episodic memory to help mobile agents complete multistep tasks. This approach uses RatSLAM as the hippocampus’s spatial navigation mechanism and episodic memory to recall the steps of a given task.

Furthermore, the researchers proposed an enhancement allowing the robot to achieve global localization within familiar areas. In their global localization module, the system addresses the “kidnapped robot” by allowing it to shut down and restart randomly within the environment. After restarting and turning on the global localization module, RatSLAM utilizes the robot’s visual input to determine whether the current location matches any mapped area information. Instead of generating new experiences as it would with new visual input, RatSLAM compares the visual information with that stored in the map. This approach aligns with scenario (i) mentioned earlier. However, this solution only works effectively when the robot restarts within a previously mapped area. As a result, the robot cannot explore distinct regions in multiple sessions when operating in global localization mode since no new experiences are created in this new environment. Consequently, it does not address the multisession scenario (ii).

While RatSLAM itself does not provide a complete solution for multisession mapping, there have been efforts to address this challenge in other non-biological SLAM frameworks (MCDONALD et al., 2013; LABBé; MICHAUD, 2014; WANG et al., 2016; LABBÉ; MICHAUD, 2018; DAOUD et al., 2018; SCHNEIDER et al., 2018; BURGUERA; BONIN-FONT, 2019; CAMPOS et al., 2021; LABBé; MICHAUD, 2022). Two notable examples are SLAMM (DAOUD et al., 2018) and ORB-SLAM 3 (CAMPOS et al., 2021), which tackle the multisession scenario by initiating a new map when tracking is lost. These solutions continuously compare new inputs with those from different stored maps and perform loop closures based on whether the information belongs to the “active” map or another map. If it belongs to another map, the two maps are merged, creating a new "active" map. This merging process involves aligning the matched inputs and integrating the information from the current map into the stored one.

Overall, previous approaches have addressed multisession mapping either in non-biological SLAM frameworks or only partially for RatSLAM, such as the work by (TANG; YAN; TAN, 2018) for scenario (i). This work focuses on an approach that mutually solves scenario (i) and the more challenging scenario (ii). However, to comprehensively deal with the multisession challenge in RatSLAM, a solution must consider the RatSLAM structures and their intrinsic relationship. For instance, reactivating a RatSLAM experience on the map requires correctly activating the visual input and the CAN related to that experience. Building multisession solutions for RatSLAM, particularly for its network, presents unique challenges.

In this context, we propose a novel approach to merge the RatSLAM Pose Cell Network, ensuring the consistency of the Experience Map and performing accurate loop closures following the merging of RatSLAM structures. Existing literature has yet to extensively explore this aspect, making our proposed solution a valuable contribution to the field. The development of the multisession solution for RatSLAM is found in Chapter 4.

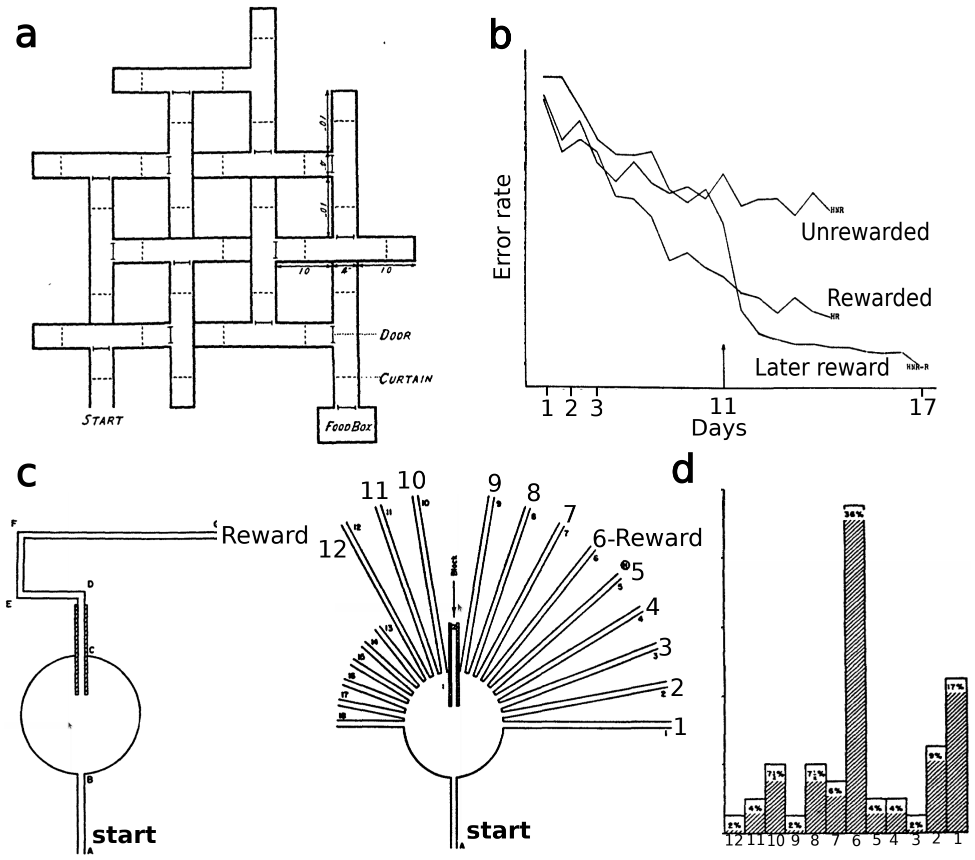
3.3 Learning Cognitive Maps for Flexible Behavior

Latent learning has been a topic of great interest in neuroscience for nearly a century. Behavioral studies have shown that this learning benefits animals in seeking rewards more quickly when familiarized with the empty environment than those with no previous experience but rewarded from the start (BLODGETT, 1929; TOLMAN; HONZIK, 1930). The initial focus of researchers was to elucidate if animals exhibit adaptive behavior through stimulus-response or cognitive map theories (BLODGETT, 1929; TOLMAN, 1948; SUTHERLAND; LINGGARD, 1982; SUTHERLAND et al., 1987; KEITH; MCVETY, 1988). They also studied the brain regions responsible for facilitating this type of learning (KIMBLE; GREENE, 1968; KIMBLE; BREMILLER, 1981; MEANS, 1969; OWEN; BUTLER, 1980; KIMBLE; JORDAN; BREMILLER, 1982). Recent studies have explored latent learning experiments in analyzing neural activity in animals' spatial and non-spatial decision-making tasks (GUO et al., 2020; BARROS et al., 2021).

Blodgett (1929) introduced the concept of latent learning through his experiments with rats in a maze. His study divided the rats into one control and two experimental groups. The control group received rewards for successfully navigating the maze from the first day, while the experimental groups were only rewarded after the third and sixth days. Blodgett observed that the control group consistently improved their performance over time, with a decreasing number of errors in subsequent trials. In contrast, the experimental groups showed no evidence of learning during the unrewarded days. However, both groups exhibited a rapid decrease in errors once rewarded, indicating that learning had taken place *latently* during the non-rewarded trials. Hence, when comparing maze running performance, latent learning refers to the significant performance improvement of a “rewarded” group compared to other groups that underwent "unrewarded" trials during the exploration phase (THISTLETHWAITE, 1951).

Tolman and Honzik made further investigations into latent learning (TOLMAN; HONZIK, 1930; TOLMAN, 1948) and showed that rats also improved learning after becoming familiar with an even more complex 14-arm T-maze (Fig. 7a,b). The rats were divided into three groups: the rewarded, unrewarded (control), and the late rewarded group. The rewarded group received food rewards at the end of the maze, leading them to learn and navigate efficiently. On the other hand, the unrewarded group received no rewards, serving as a control group to compare against the other two groups. The late rewarded group received no rewards for the first ten days but was later reinforced with food rewards. Similar to the findings observed by Blodgett, the late-rewarded group demonstrated notable improvement in locating the reward within the maze despite no evidence of learning before the introduction of rewards. Furthermore, the researchers conducted experiments in which rats were guided to follow a specific route leading to a reward location (Fig. 7c). Subsequently, new accessible routes were introduced after blocking the learned one. Interestingly, most animals chose the arm whose direction and distance were closest to the former reward location (Fig. 7d). These experiments supported the theory that the animals acquired a *cognitive map* and could internally compute novel routes using their knowledge of the distance and direction to the reward location.

Figure 7 – Tolman's experiments in latent learning.



Source: Adapted from (TOLMAN, 1948)

In addition to the evidence supporting the formation of cognitive maps in latent learning experiments, research suggests that replay mechanisms (PAVLIDES; WINSON, 1989; KARLSSON; FRANK, 2009) play a crucial role in the consolidation and utilization of these maps (ÓLAFSDÓTTIR; CARPENTER; BARRY, 2016; POULTER; HARTLEY; LEVER, 2018; MOMENNEJAD, 2020; SCLEIDOROVICH et al., 2020; DIEKMANN; CHENG, 2023). One is awake replay (KARLSSON; FRANK, 2009), where place cells where observed to reactivate during spatial navigation, both in the forward direction (before actual movement) and in the reverse direction (after task completion). In a recent study, Guo and colleagues reported a potential role for replay from rest periods in consolidating learning between different places in an environment (**??**). In the absence of rest, animals' spatial representations tend to disintegrate from the spatial environment's structure, a phenomenon not observed in animals that take regular rest breaks.

Latent learning experiments yet raise a crucial question about how cognitive maps are encoded in the brain, particularly concerning animals exhibiting flexible behavior in response to rewards. A prominent explanation is provided by the predictive map hypothesis, which asserts that animals acquire the ability to anticipate long-term rewards by predicting their future locations (or states) based on their current position (STACHENFELD; BOTVINICK; GERSHMAN, 2017; GERSHMAN, 2018). According to this hypothesis, place cells encode predictions of future states, which illustrates why the firing patterns of place cells are modulated by factors such as obstacles and environmental topology (STACHENFELD; BOTVINICK; GERSHMAN, 2017). Furthermore, the predictive map perspective is formulated within the Reinforcement Learning paradigm, wherein the Successor Representation algorithm (DAYAN, 1993) is employed to capture the underlying mechanisms of predictive maps.

Several studies have explored Successor Representation (SR) as an alternative framework for understanding flexible behavior. Gershman (2018) provides a comprehensive review of behavioral and neural studies that support the SR structure. Russek et al. (2017), de Cothi et al. (2022), Ducarouge e Sigaud (2017) investigate the SR as a model of behavioral flexibility, comparing it with other RL algorithms such as model-based and model-free approaches. Notably, Ducarouge e Sigaud (2017) revisit experiments on latent learning using the SR in Blodgett's maze (BLODGETT, 1929).

However, a significant challenge for the RL and thereafter for the SR lies in defining states for the algorithm to predict their relationships accurately. This may be necessary in situations where the explicit definition or pre-definition of states in an environment is challenging or impractical. Moreover, states can be acquired from raw data, which may introduce complexity if they are sparse or have high dimensionality. One approach involves pre-defining states before simulations, which requires modeling the dynamics of the environment as an additional task. Otherwise, an alternative method is to utilize State Representation Learning (SRL) (LESORT et al., 2018) techniques, which aim to autonomously learn and represent states based on the available sensory information or observations.

This work presents a framework integrating two fundamental processes, mapping and learning, within the CoBeL-RL (DIEKMANN et al., 2023) framework. Our extension to CoBeL-RL allows RL agents to learn in unfamiliar environments using RatSLAM as an SRL. RatSLAM allows the agent to differentiate between different locations in the environment, relying on its pose cells network. This capability allows the agent to learn from an unknown condition of the states of the environment, filling a gap in existing works that assume that learning starts totally or partially with pre-defined states. Moreover, by applying this framework to the study of latent learning, we aim to gain novel insights into the mechanisms that underlie flexible behavior in spatial navigation and cognitive maps.

3.4 Experimental Designs in Latent Learning

A relevant question regarding latent learning is whether this learning depends on specific environmental or behavioral conditions and constraints during the learning process. An example is the use of doors to guide animal trajectory and modify their exploration strategy within mazes, as explored by Tolman (TOLMAN, 1948). Daub (1933) analyzed this impact on maze performance. The animals in the experiment were divided into four distinct groups: two groups were exposed to an environment featuring doors, further categorized into latent and standard subgroups, while the remaining two groups were devoid of doors, consisting of latent and standard subgroups. In the maze pre-training phase, latent door and no-door groups were allowed to freely explore a 14-T maze for eight hours daily without any food or restrictions imposed by the doors. Thus, the animals had unrestricted movement during this period. Each animal received a daily trial to run the maze in the reward phase. In this evaluation period, the doors were closed behind the animals for the door groups every time they passed, preventing them from retreating. The results indicated a significant difference in both time and error scores between the latent learning and the standard groups, with the latent without doors showing the lowest error scores. Interestingly, the presence of doors did not significantly impact scores for either latent group. Nevertheless, it is important to mention that both latent learning groups had the same pre-exposure regime, i.e., they explored the environment without doors. This condition might have influenced them to exhibit similar learning scores.

Latent learning experiments also offer various design prospects that may impact animals' understanding of spatial relationships within their environment and how they build cognitive maps. A literature review by Thistlethwaite (1951) categorized different types of latent learning designs employed during the maze pre-training phase. The first type involved a single attempt to explore the environment. The second type included brief periods of exploration and living in an empty maze, followed by introducing a relevant goal object and subsequent test trials. The third type includes trials where rats navigated a maze with pre-satiated target objects, followed by additional tests after inducing hunger or thirst. Lastly, the fourth type consisted of tests conducted with hungry or thirsty rats in a maze containing relevant and irrelevant goal objects. Notably, the reports reviewed consistently demonstrated the presence of latent learning across all these experimental designs.

Within this study's scope, we focus on the first two types of latent learning design experiments. The single attempt (or single trial) exploration is characterized by establishing a relationship between the goal location and changing the animal's state during exposure, such as removal from that location and confinement upon arrival. Moreover, building a connection to the goal location can also be achieved by placing the animal at the goal location. In contrast, the second type, time-fixed exploration, does not involve such a connection to the goal location, as it does not provoke changes in the animal's state at this place. Instead, the animal is removed from its current position once the allotted time limit expires.

Karn e Jr. (1946) conducted experiments to investigate differences in maze performance over different experiment designs. Specifically, they examined the influence of various pre-training experiments on latent learning to assess rat performance in finding food using a Dashiell maze. The animals were exposed to different conditions, including familiarity with handling, confinement, and exploratory type one and two experiences. The results demonstrated that the pre-training design closest to the rewarded task, i.e., single-trial exploration, had a more pronounced effect on improving task performance. Furthermore, exploratory maze experiences significantly impacted animals' latent learning, overcoming the effects of pre-training without exploration.

In an additional investigation by Sutherland e Linggard (1982), the latent learning performance of rats was assessed through different types of exposure before training in the Morris water maze (VORHEES; WILLIAMS, 2006). The rats were divided into three groups: one group received exposure to the correct platform location, another was exposed to an incorrect location, and the third was exposed to the correct location but in a different room (naive group). During the training trials, each rat had a brief 30-second swimming period before being placed on the platforms. Subsequently, they were given a 10-minute timeframe to complete the task during the reward trials. Results demonstrated that the first group quickly learned the task, while the second group took longer to find the platform than the naive group. However, once learned, the second group outperformed the naive group.

The findings from (KARN; JR., 1946; SUTHERLAND; LINGGARD, 1982) indicate that pre-training strategies that closely resemble the rewarded task, such as single trial exploration, significantly influence rat performance more than time-fixed exploration in latent learning. Furthermore, these findings emphasize the role of exploration in facilitating efficient navigation toward a goal during cognitive map acquisition. On the other hand, lack of exploration or exploration in different environments during pre-training does not significantly affect latent learning.

In conclusion, the design of latent learning experiments plays a crucial role in influencing the rate at which animals learn when a reward is introduced. In this study, we investigate the underlying reasons for these differences observed in latent learning. We hypothesize that the variations in exploration designs result in distinct learned representations of the environment, ultimately impacting the learning speed between single trial and time-fixed exploration. Consistent with the earlier findings, the studies emphasize the necessity of some exploration to observe the latent learning effect.

3.5 Summary

This chapter focuses on the challenges associated with navigating and learning in unfamiliar environments. We introduce the RatSLAM, a computational cognitive model that tackles the complex task of mapping and merging maps from multiple sessions. Furthermore, we delve into latent learning and examine current studies in reinforcement learning setups to shed light on how cognitive maps may be encoded in the brain. In addition, we explore studies that show the impact of various exploration strategies during the unrewarded phase on learning speed. The findings from these works indicate that exploration strategies closely aligned with the critical rewarded run lead to significant improvements in learning compared to uniform unrewarded exploration or limited exploration.

4 A Multisession Approach for RatSLAM

This chapter presents our proposed approach's methodology and results to enhance RatSLAM mapping capability across multiple SLAM sessions. We address the type (ii) multisession solution, where the robot initiates mapping in a new location using its reference frames and coordinates, subsequently integrating this map with previously stored ones. We have applied this method to diverse environments, ranging from small virtual to large-scale real-world datasets.

Our approach assumes the robot has already created and saved a map as a RatSLAM structure. This structure includes the Local View Cells (LVC), Pose Cells Network (PCN), and Experience Map (EM). This saved map is the *loaded map* of the environment. When the new session begins, the agent loads this map into its memory but assumes no prior knowledge about its possible location on this map.

During the current session, the agent starts building a new map, also known as the *partial map* of the environment. Our method involves merging partial and loaded maps if the robot crosses any location earlier mapped in the loaded maps. Hence, the merging occurs when the loaded and partial maps intersect at a specific location in the physical environment. The two RatSLAM structures are combined through the merge mechanism, resulting in a unified representation that includes a single LVC, PCN, and EM to represent the environment accurately. However, if the robot begins the session at a known location within the loaded map, the algorithm activates the corresponding experience within the loaded map. This process is equivalent to multisession solution type (i). We elaborate on this merging process in the subsequent sections.

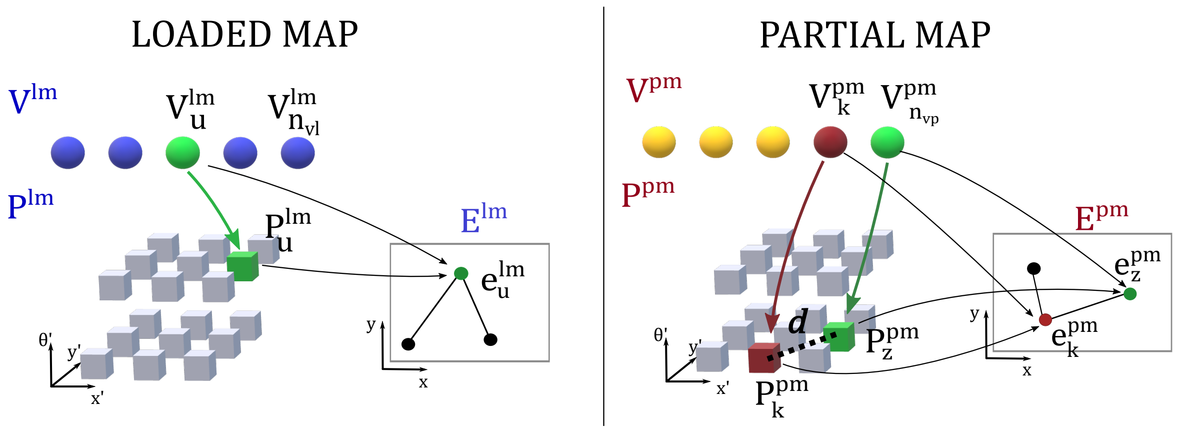
4.1 Methodology

The merging process is triggered when the agent generates a template in the partial map that matches a previously saved template in the loaded map (Fig. 8), which means the robot has visited a common location in both mapping sessions. To compare these templates, we employ the same procedure used in RatSLAM, i.e., where a new template from the partial map is compared to all the templates in the loaded map.

We formalize that both loaded and the partial maps consist of RatSLAM's structures (see Fig. 3), where V, P, E, lm and pm stand for Local View Cell, Pose Cell Network, Experience Map, loaded map and partial map, respectively. The template of the partial map matches a template from the loaded map $V^{\text{pm}}_{n_{vp}} = V^{\text{lm}}_{u}$ (Fig. 8), where n_{vp} is the number of templates of V^{pm}. Furthermore, the views $V^{\text{pm}}_{n_{vp}}$ and V^{lm}_{u} are linked to the center of the activity packet, displayed as the green cubes, P^{pm}_{z} and P^{lm}_{u}, respectively. However, even though the templates represent the same place, their activity packets may activate different coordinates (x', y', θ') in the respective PCN, P^{pm} and P^{lm}. Similarly, for the EM, the templates and activity packets are associated with experiences e_z^{pm} and e_u^{lm} (Fig. 8, green nodes), where these experiences may have different poses coordinates.

Relevant information for merging also depends on the last template activated in the partial map before meeting the condition for merging, corresponding to the state $k = n_{vp} - 1$. Note that the linked centers of the activity packet P_z^{pm} and P_k^{pm} are spatially separated by a distance d. Likewise, their associated experiences e_z^{pm} and e_k^{pm} have link information (Eq. 2.12) that encodes the distance information between their poses.

Figure 8 – Condition for the merging procedure between two RatSLAM structures.



Source: Designed by the author and adapted from (MENEZES et al., 2023)

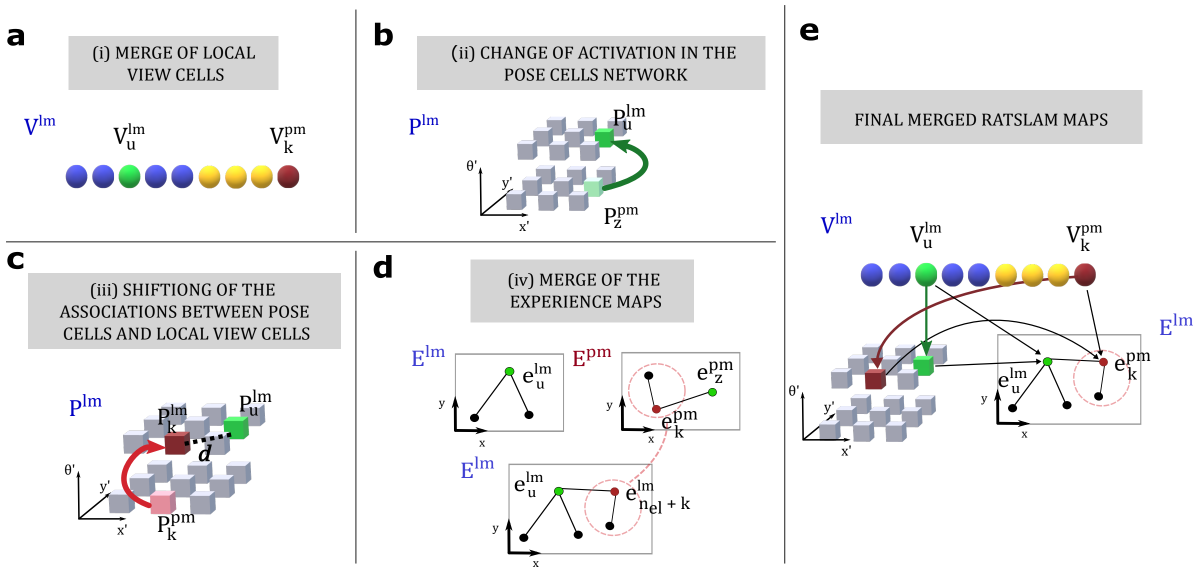
During the merging process, the partial RatSLAM map structures and their relationships are inserted in the loaded map in four operations (Fig. 9): (i) the LVC are merged (Fig. 9a), (ii) the matching template in the partial map $V_{n_{vp}}^{pm}$ is linked to a new PCN activity packet location, which corresponds to a shift (Fig. 9b), (iii) the associations between all LVC and PCN are shifted by the same amount (Fig. 9c), and (iv) the EMs are merged (Fig. 9d). The following subsections provide a more detailed explanation of these operations. Once the merge procedure is complete, a single RatSLAM structure is obtained (Fig. 9e) and continues the mapping session.

4.1.1 Merging Local View Cells

The purpose of merging LVC is to combine the templates from partial and loaded maps into a unified LVC structure. All templates from the partial map are concatenated to the LVC of the loaded map, except for the most recently acquired template already included in the loaded structure.:

$$V^{\text{lm}} = [V^{\text{lm},1}, ..., V^{\text{lm},n_{vl}}, \mathbf{V}^{\text{pm},\mathbf{1}}, ..., \mathbf{V}^{\text{pm},\mathbf{n_{vp}-1}}] \tag{4.1}$$

Figure 9 – Merging partial and loaded RatSLAM maps.


a
(i) MERGE OF LOCAL VIEW CELLS
b
(ii) CHANGE OF ACTIVATION IN THE POSE CELLS NETWORK
c
(iii) SHIFTIONG OF THE ASSOCIATIONS BETWEEN POSE CELLS AND LOCAL VIEW CELLS
d
(iv) MERGE OF THE EXPERIENCE MAPS
e
FINAL MERGED RATSLAM MAPS

Source: Designed by the author and adapted from (MENEZES et al., 2023)

where n_{vl} is the number of templates of V^{lm}.

In the last stage of the LVC merge, the template that represents the current robot's view scene V_u^{lm} is activated in the V^{lm} (Fig. 9, green sphere).

4.1.2 Pose Cells Network Activation

Once the LVC is merged, the activated template is V_u^{lm}. Thus, to correctly activate the PCN units associated with this template, an injection of activity should occur at the coordinates of P_u^{lm}. Before the merge procedure, the last activated packet in the partial map was P_z^{pm} (Fig.9b, light green cube). Therefore, a change of activity from P_z^{pm} to P_u^{lm} is necessary and can be observed as a shift of activity in the P_u^{lm} (Fig. 9b, green arrow). The difference of coordinates between P_z^{pm} and P_u^{lm} is defined as:

$$\begin{aligned} \Delta x' &= x'_p + x'_l; \\ \Delta y' &= y'_p + y'_l; \\ \Delta \theta' &= \theta'_p + \theta'_l. \end{aligned} \tag{4.2}$$

where (x'_p, y'_p, θ'_p) and (x'_l, y'_l, θ'_l) are the coordinates of activity packets P_z^{pm} and P_u^{lm}, respectively.

4.1.3 Shifting Association between LVC and PCN

To ensure consistency in the loaded map, all associations between LVC and PCN in the partial map must be updated to match their new positions in PCN. As an example, we consider the penultimate view in the partial map. P_k^{pm} must be shifted to keep the previous distance d to P_u^{lm} (Fig. 9c, red arrow). The transformation function that shifts the activity packet P_k^{pm} to P_k^{lm} is defined as $f()$:

$$\begin{aligned} &f(x'_u, y'_u, \theta'_u) = (x'_s, y'_s, \theta'_s);\\ &x'_s = (x'_u + \Delta x')(\text{mod } n_{x'});\\ &y'_s = (y'_u + \Delta y')(\text{mod } n_{y'});\\ &\theta'_s = (\theta'_u + \Delta \theta')(\text{mod } n_{\theta'}). \end{aligned} \tag{4.3}$$

where (x'_u, y'_u, θ'_u) are the coordinates of P_k^{pm}. The (x'_s, y'_s, θ'_s) are the shifted coordinates of P_k^{lm} in the loaded map.

Once this shifted operation is carried out, the excitatory links β^{lm} are updated as follows:

$$\beta^{\text{lm}}_{n_{vl}+i, f(x',y',\theta')} = \beta^{\text{pm}}_{i,x',y',\theta'}, i \in \{1, ..., n_{vp} - 1\}. \tag{4.4}$$

where (x', y', θ') are the coordinates of the energy packets from P^{pm}.

4.1.4 Merging Experience Maps

Before merging the experiences of E^{pm} into E^{lm}, it is necessary to establish a consistent transformation between the experiences in the partial map and those in the loaded map. Before the merge, the experience e_z^{pm} represented the actual pose of the robot and was linked to e_k^{pm} in E^{pm}, as shown in Fig. 8. This e_z^{pm} is equivalent to e_u^{lm} in the loaded map. The experiences e_z^{pm} and e_k^{pm} encode the final and penultimate poses of the robot in the partial map.

The topological-metric relation between e_k^{pm} and e_z^{pm} must be kept between e_k^{pm} and e_u^{lm} after the merge process by applying the same transformation on e_z^{pm} and e_k^{pm}. e_k^{pm} after the transformation is denoted as $e^{\text{lm}}_{n_{el}+k}$ (see notation in Eq. 4.8) and it is shown in the merged EM E^{lm} in Fig. 9d where the dashed red line indicates the transformation.

This transformation involves a combination of translational and rotational operations in two dimensions. Let $\mathbf{p}_l = [x_l, y_l, \theta_l]^T$ and $\mathbf{p}_p = [x_p, y_p, \theta_p]^T$ be the poses of the equivalent experiences e_u^{lm} and e_z^{pm}, respectively. The function $\mathbf{t}$ transforms the pose $\mathbf{p}_p$ to the pose $\mathbf{p}_l$, i.e. $\mathbf{t}(x_p, y_p, \theta_p) = x_l, y_l, \theta_l$, and is defined as follows:

$$\mathbf{t}(x_p, y_p, \theta_p) = [H(x_p, y_p), \theta_p + \Delta\theta]^T. \tag{4.5}$$

where $\Delta\theta = \theta_l - \theta_p$ and $H(x_p, y_p)$ is defined by:

$$H(x_p, y_p) = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} x_p - \Delta x \\ y_p - \Delta y \end{bmatrix}. \tag{4.6}$$

where $\Delta x = x_l - x_p$ and $\Delta y = y_l - y_p$. Once function $\mathbf{t}$ is defined, it must be applied on the poses of E^{pm} experiences so they could be inserted on the merged E^{lm}, which is carried out through operation T, defined below. In addition, operation T also changes the information of the new experiences in E^{lm} with their correspondent template and energy packets according to operations (i) and (iii) loaded map's structures V^{lm} and P^{lm}.

$$\begin{aligned} e_{n_{el}+i}^{\text{lm}} &= T(e_i^{\text{pm}}), i = 1, ..., n_{ep} - 1; \\ &= \{V^{\text{lm}, i+n_{vl}}, P_{f(x', y', \theta')}^{\text{lm}, i+n_{vl}}, \mathbf{t}(\mathbf{p}^{\text{pm}, i})\}. \end{aligned} \tag{4.7}$$

where e^{pm} is an experience from E^{lm}. The n_{el} and n_{ep} are the number of experiences in E^{lm} and E^{pm} maps, respectively. It is important to mention that e_z^{pm} is not added in E^{lm} because it is already equivalent to e_u^{lm}. Then, the new experiences are inserted on the merged E^{lm}.

$$E^{\text{lm}} = [e_1^{\text{lm}}, ..., e_{n_{el}}^{\text{lm}}, \mathbf{e}_{\mathbf{n_{el}+1}}^{\text{pm}}, ..., \mathbf{e}_{\mathbf{n_{el}+n_{ep}-1}}^{\text{pm}}] \tag{4.8}$$

The last step iv) is to connect, through links, the added nodes on the E^{lm} as follows:

$$l_{i+n_{el}, j+n_{el}} = \{\Delta \mathbf{p}^{(i+n_{el})(j+n_{el})}, \Delta t^{i,j}\} \tag{4.9}$$

$i = 1, ..., n_{ep} - 2$ and $j = 2, ..., n_{ep} - 2$, which is similar to (2.12). As e_z^{pm} is not inserted its equivalent, e_u^{lm}, has to be connected to $e_k^{\text{pm*}}$ through (4.9) with $i + n_{el}$ being k and $j + n_{el}$ is u.

4.1.5 Merge Algorithm

The merge algorithm is outlined in the Algorithm 1, highlighting the equations that make up each step. Complexity analysis can consider the match and the merge routines separately. The computational complexity of the merge process is determined by the matching of a template in the partial map of size n_{vp} with a template of size n_{vl} from previous sessions stored in κ, the computational complexity can be given by:

- Matching routine: depending on the search implementation, it could be $\kappa \times O(1)$, $\kappa \times O(log(n_{vl}))$ or $\kappa \times O(n_{vl})$;
- MergingLocalViewCells(line 2): $O(n_{vp})$;
- ComputeDeltasPCN(line 3): $O(1)$;
- P^{lm} updating (line 4): $O(1)$;
- ShifitingAssociationLvcPcn(line 5): $O(n_{vp})$;
- ComputeTransformation (line 6): $O(1)$;
- MergeExperiencesMaps (line 7): $O(n_{ep})$.

Algorithm 1 Merge of RatSLAM Structures

```
1: procedure MERGE($V^{lm}, V^{pm}, P^{lm}, P^{pm}, E^{lm}, E^{pm}$)
2:     $V^{lm} \leftarrow$ MergingLocalViewCells()                    ▷ Eq. 4.1
3:     $\Delta x', \Delta y', \Delta\theta' \leftarrow$ ComputeDeltasPCN()           ▷ Eq. 4.2
4:     $P^{lm}_{current} \leftarrow P^{lm}_{u}$                                 ▷ PCN activation
5:     $P^{lm} \leftarrow$ ShifitingAssociationLvcPcn()               ▷ Eq. 4.4
6:     t() $\leftarrow$ ComputeTransformation()                     ▷ Eq. 4.5
7:     $E^{lm} \leftarrow$ MergeExperiencesMaps()                    ▷ Eq. 4.7,4.8,4.9
```

In summary, the merge algorithm's complexity stays within linear behavior when it comes to **Local View Cells** (in partial and loaded maps) and **Experience Map** sizes.

4.2 Experimental Setup

This study explores four different environments, and for each of them, datasets of video streams or image frames were collected from tours conducted by real and virtual robots. The four environment datasets consist of i) videos generated by a virtual robot during an ellipse-shaped tour, referred to as the *Virtual Tour* dataset; ii) videos generated by a real robotic platform touring inside a research lab, known as the *Lab Tour* dataset; iii) frames extracted from the *"iRat"* Australian dataset; and iv) frames extracted from *The New College Vision and Laser dataset*. The latter two were employed to validate the OpenRatSLAM implementation (BALL et al., 2013). Each environment is further detailed in the subsequent subsections.

To evaluate the proposed multisession approach, the maps from single-session and multisession are compared using the Iterative Closest Point (ICP) algorithm

(BESL; MCKAY, 1992). ICP addresses the registration problem by iteratively finding a transformation matrix that aligns the two maps as closely as possible. To evaluate the accuracy of the transformation matrix, ICP computes the *root mean square errors* (RMSE) between corresponding node distances in both maps. The iterations stop when the RMSE falls below a defined threshold, or the algorithm reaches the maximum number of iterations. By providing the RMSE over the distances of corresponding nodes, ICP yields a single value that evaluates the overall trajectory of the multisession and single-session maps, with an RMSE of 0 indicating a perfect match. The Libicp algorithm (GEIGER; LENZ; URTASUN, 2012) is utilized for these purposes in this article.

The RatSLAM algorithm requires a specific set of parameter values for each environment (BALL et al., 2013; MILFORD; WYETH, 2008). These parameters are required in all the main structures, i.e. LVC, PCN, and EM. The parameter values for each environment are displayed in Tab. 2. In both the *"iRat"* and New College datasets, odometry information is obtained from the robot's wheel encoders, so the Visual Odometry parameters were not used.

The multisession RatSLAM source code was implemented using Python 3.6 and is available at <https://zenodo.org/badge/latestdoi/568248424>.

4.2.1 *Virtual Tour* experiment

In the *Virtual Tour* experiment, the robot takes an ellipse-shaped tour in a virtual environment (Fig. 10). The environment and the robot were modeled in an earlier version of CoBeL-RL (DIEKMANN et al., 2023).

This experiment consists of two mapping sessions. In the first session, the robot completes three-quarters ($3/4$) lap through the environment (Fig. 10b (blue line)). When the robot reaches the end of this path (blue diamond), the experience map is saved. In the second session, the virtual agent loads the previously saved map but starts mapping from a new location (yellow cross) within the environment, creating the partial map. The robot performs almost three complete laps (yellow line) and eventually travels on the same path as in the first session.

The merge between the partial map from the second session and the loaded map from the first session takes place when the agent encounters a view that is already stored in the loaded map (Fig. 10b, red circle). Once the merge occurs, the virtual robot continues mapping using the merged RatSLAM structure until the end of the second session.

It is important to note that the video frames that generated the map in the first session are included in the video frames used in the second session. Specifically, the video from the first session is embedded in the video stream used during the second session. As a result, when the agent reaches the merge point in the second session,

Table 2 – Parameter of RatSLAM's to each tested environment.

name	value			
	Virtual Tour	Lab Tour	iRat	New College
# Visual Odometry				
vtrans_image_x_min	0	80	-	-
vtrans_image_x_max	256	560	-	-
vtrans_image_y_min	0	240	-	-
vtrans_image_y_max	64	360	-	-
vtrans_scaling	10.0	10	-	-
vtrans_max	0.1	0.1	-	-
vrot_image_x_min	0	80	-	-
vrot_image_x_max	256	560	-	-
vrot_image_y_min	0	240	-	-
vrot_image_y_max	64	360	-	-
camera_fov_deg	360	50	-	-
camera_hz	1	1	-	-
# Local View module				
vt_panoramic	0	0	0	1
vt_match_threshold	0.05	0.04	0.035	0.059
vt_shift_match	5	4	4	4
vt_step_match	2	1	1	10
vt_normalisation	0.5	0.5	0.5	0.5
vt_active_decay	1.5	1.0	1.0	1.5
template_x_size	128	80	50	60
template_y_size	32	60	30	10
# Pose Cell module				
pc_vt_restore	0.05	0.05	0.05	0.05
pc_dim_xy	80	80	11	80
pc_vt_inject_energy, δ	0.06	0.5	0.6	0.06
pc_dim_th	35	36	36	35
exp_delta_pc_threshold	4.0	1	2.0	4.0
pc_cell_x_size	1	1	0.015	1.0
# Experience Map module				
exp_loops	250	250	20	20
exp_initial_em_deg	140	90	140	140

the subsequent frames are the same frames collected in the first session. Therefore, RatSLAM should not create new experiences as the agent traverses the path covered in the first session (blue path).

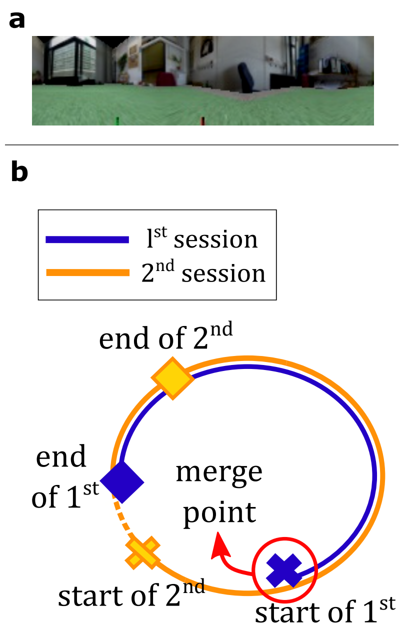
Furthermore, after the merge, a section of the path between the end of the first session and the beginning of the second session (the dashed yellow line between the blue diamond and yellow cross) has not been mapped. The agent in the second session should map this section to close the loop of the ellipse path fully.

4.2.2 *Lab Tour* experiment

The *Lab Tour* experiment involves mapping a research laboratory (Fig. 11a) using a robot platform called RoboDeck[1] (Fig. 11b). The RoboDeck platform has a monocular camera with a resolution of 640×480. In this experiment, only the robot's camera was

[1] http://www.xbot.com.br/

Figure 10 – Virtual Tour environment setup.



Source: Designed by the author and adapted from (MENEZES et al., 2023)

used to capture the video streams, and the odometry information was extracted using visual odometry. The video streams for this experiment were obtained by manually driving the RoboDeck robot platform on a room tour. The robot follows a rectangular trajectory resembling an eight shape (Fig. 11c).

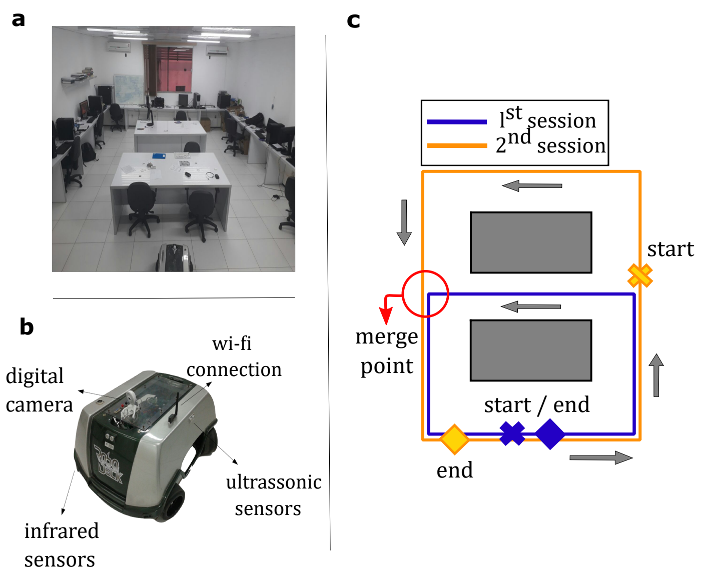
In the first session, the robot completes two counterclockwise laps around the small rectangle, starting at the location marked by the blue cross and ending at the blue diamond (Fig. 11c, blue line). In the second session, the robot completes nearly two counterclockwise laps around the large rectangle, starting at the location marked by the yellow cross and ending at the yellow diamond (Fig. 11**c**, yellow line).

The merge between the maps is expected to occur at the point where the two paths first meet (Fig. 11c, red circle). After the merge, a final loop closure is expected when the robot returns to the start point of the second session (yellow cross).

4.2.3 *iRat* experiment

The multisession approach used the *iRat 2011 Australia dataset* to validate the method with more than two mapping sessions, focusing on consistency in more than two sessions. The dataset was collected while exploring an outdoor road tour by a small mobile robot named *iRat*, which resembled a large rodent in size and shape (BALL et al., 2013). The robot was equipped with an overhead camera and dead reckoning sensors to capture images and odometry data.

Figure 11 – Lab Tour experiment setup.



Source: Designed by the author and adapted from (MENEZES et al., 2023)

The complete dataset consists of a video approximately 16 minutes long (BALL et al., 2013), during which the robot explores the environment by moving without any specific pattern along the roads. However, only a portion of the dataset frames corresponding to five mapping sessions was used for the experiment. The first four sessions involve internal laps within the environment. The last session corresponds to the external lap (Fig. 12b) and therefore merges the four internal maps into a single one when it travels through their common areas.

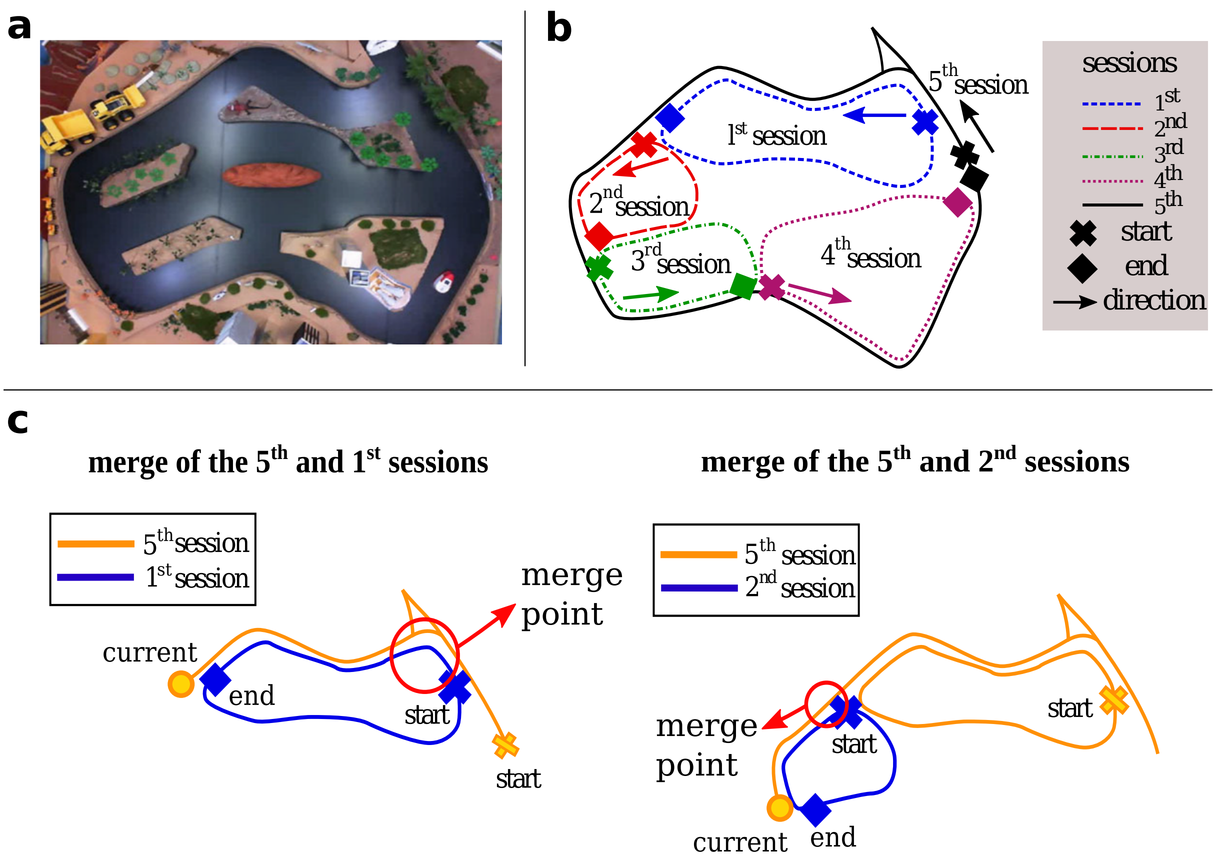
The map from one mapping session is transferred and used as the loaded map in the next session. The two maps can be merged once the partial map session overlaps with the loaded map. This merging process is then repeated in subsequent mapping sessions. Figure 12c illustrates the merge points between the fifth and first sessions (red circles) and the fifth and second sessions, respectively.

4.2.4 *New College* experiment

The New College Vision and Laser dataset is a comprehensive dataset collected from a robot that completed multiple outdoor loops around the New College campus in Oxford (SMITH et al., 2009). The dataset includes $360°$ images (Fig. 13a), odometry, and laser scan data. Similar to the *iRat* dataset, the New College dataset has been used to validate the openRatSLAM implementation (BALL et al., 2013). Given its complexity, this experiment aims to validate the multisession approach using full-scale long-term data.

The experiment consists of three mapping sessions. The first two sessions cover different areas of the environment (Fig. 13b, blue and green paths). The goal of the third

Figure 12 – The *iRat* experiment setup.



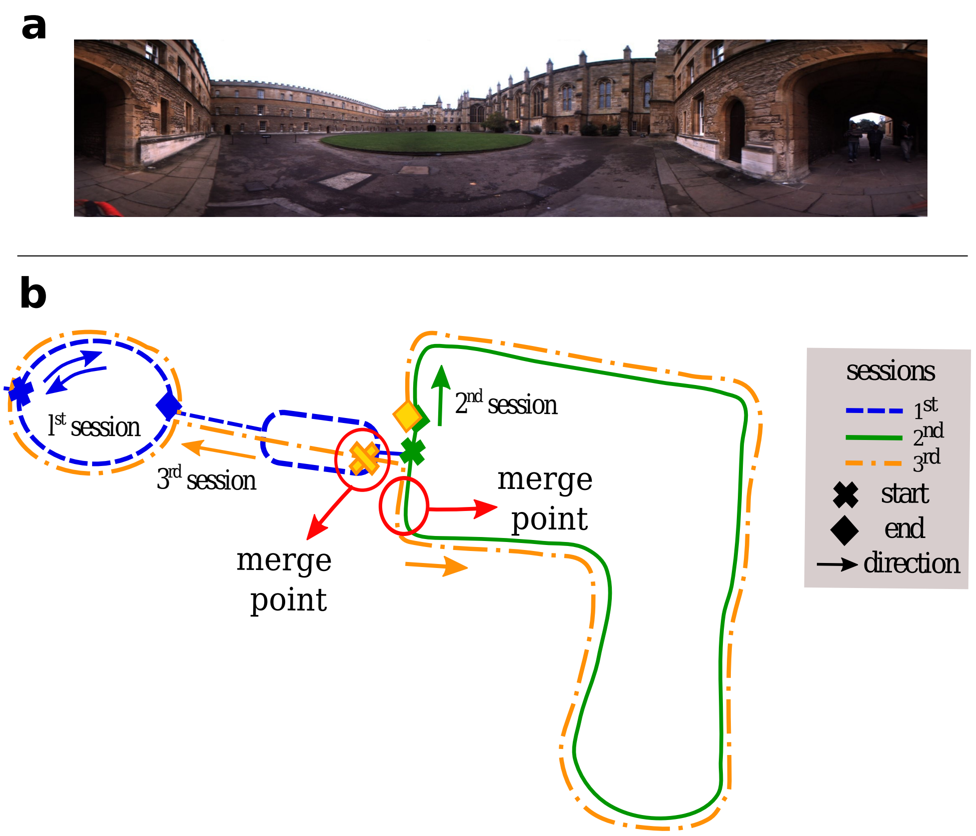
Source: Designed by the author and adapted from (MENEZES et al., 2023)

session is to merge the two loaded maps into a single map (yellow path). Additionally, this session introduces a significant amount of new visual information to the mapping. This additional input will test the approach's ability to maintain consistent mapping after the merge operations.

In the first session, the robot completes three clockwise laps inside the ellipse-shaped area (upper blue arrow). Subsequently, the robot moves towards the intermediate area between the first and second sessions and returns to the ellipse area to complete its fourth lap, this time in a counterclockwise direction (bottom blue arrow). In the second session, the robot explores different regions of the environment, completing a total of two clockwise turns.

The third mapping session begins at a position within the map from the first session (Fig. 13, yellow cross). The robot traverses the entire environment in counterclockwise laps. In particular, the robot's last counterclockwise lap within the area corresponding to the second session adds novel information for the mapping.

Figure 13 – The New College Oxford Experiment Setup.



Source: Designed by the author and adapted from (MENEZES et al., 2023)

4.2.5 RatSLAM's parameter regularization

The RatSLAM's parameter tuning can become challenging, time-consuming, and resource-consuming for new environments, especially those vastly different from the datasets experienced in validating RatSLAM (BALL et al., 2013). The *Lab tour* dataset fits this problem. In this section, a new RatSLAM parameter tuning method is presented.

In the manual tuning algorithm proposed by Ball et al. (2013), only the *vt_match_threshold* and *pc_vt_inject_energy* parameter should be adjusted. The remaining parameters should be set similarly to one example in their work. These examples support parameters for large real-world, 360 degrees, and controlled small environments. However, for a different environment, e.g., a virtual environment, the default values for the remaining parameters are not guaranteed to work appropriately.

In addition, the manual tuning algorithm assumes that the velocities parameters are correctly adjusted before the tuning of the *vt_match_threshold* and *pc_vt_inject_energy*. To check the correct values for the Visual Odometry parameters, the rate $R = \dot{T}/\dot{E}$ should be closer to 0.5 or 1.0, where $\dot{T}$ is the number of visual templates and $\dot{E}$ is the number of experiences. After tuning the Visual Odometry parameters, the *vt_match_threshold* is tuned inside a loop function to keep the R rate between 0.5 and 1.0. Finally, the *pc_vt_inject_energy* is tuned inside another loop to make RatSLAM closes a loop correctly in the environment.

The tuning process of RatSLAM by (MENEZES et al., 2020) is formulated as an optimization problem that considers the values of the parameters, the generated, and the ground-truth map. First, the Irace package automatically generates a candidate parameter set for the RatSLAM according to pre-defined ranges of values for each parameter. For each parameter set file, RatSLAM builds an EM, named generated map. Then, the deviations between the generated map and the ground truth map (obtained from the robot's odometry) are evaluated with the ICP algorithm, resulting in a residual error value. Finally, Irace considers these errors to select candidates for the next iteration and generate new combinations of parameters until the error reaches a stopping criterion. The objective function is described as follows (MENEZES et al., 2020):

$$p^* = \text{argmin}_{p\in\mathbf{P}} \sum_{k=1}^{N_V} er^{(p,v_k)} \tag{4.10}$$

that minimizes the residual error, $er^{(p,v_k)}$, between the generated map and the ground truth, computed by ICP over the 2D coordinates points. Each mapping is generated by the RatSLAM using a parameter set, p, and the input video stream v_k taken from N_V input files.

Since the ICP computes the mean squared error of the distances between the ground truth and generated map, the closest match between them leads to an error closer to 0, and the higher error variation will depend on the maximum distortion of the generated map. Therefore, the ICP can fairly represent the perfect match of the maps when the error between them is 0, but it cannot precisely inform their similarities if the error is higher than 0.

The tuning method in (MENEZES et al., 2020) is assumed to correct the *pc_vt_inject_energy* parameter if the ground truth map has a loop closure. This is because the *Irace* will positively consider parameters that generate maps similar to the ground truth. However, as the *Irace* only considers the similarity between the two maps, it does not take into account the R rate as proposed in (BALL et al., 2013), which recommends a generated EM with a R rate $\in [\alpha, \beta]$: commonly $\alpha = 0.5$ and $\beta = 0.9$. To overcome these issues, the new proposed objection function is given as follows (GOMES et al., 2022):

$$\begin{aligned} p^* =_{p\in\mathbf{P}} \sum_{k=1}^{N_V} & er^{(p,v_k)} \times (1 + \max{(R^{(p,v_k)} - \alpha, \beta - R^{(p,v_k)}, 0)}) \times \\ & \log_{10}(N_x + 10) \end{aligned} \tag{4.11}$$

where $(1 + \max(R^{(p,v_k)} - \alpha, \beta - R^{(p,v_k)}, 0))$ increases the error only if R rate is out of the recommended interval $[\alpha, \beta]$. The $\log_{10}(N_x + 10)$ penalizes generated maps with the higher number of nodes N_x. Hence it is assumed that the *Irace* prioritizes those parameters that maintain the R rate in the desired value, leading to configurations that generate maps with a minor number of points but are better distributed over the map.

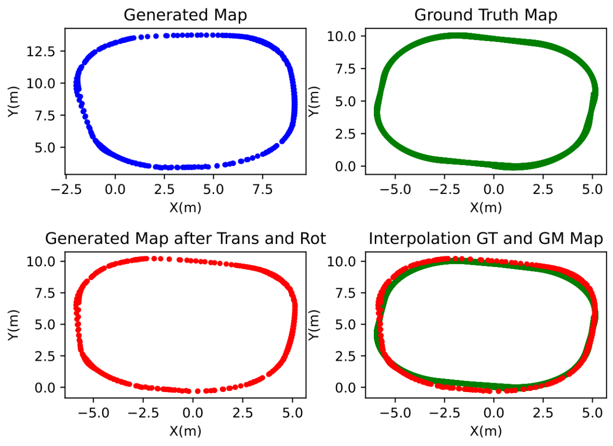
An experiment is proposed to evaluate the robustness of the RatSLAM tuning method, focusing on the generalization capacity for different laps performed by a robot within a virtual environment that resembles the *Lab Tour*, i.e., an “eight”-shape environment. The experiment consists of tuning RatSLAM for specific laps (training stage) and assessing the mapping quality in the different laps as well as the entire environment (testing stage). This type of experiment is also justified because tuning time over smaller laps, describing simpler geometric figures, is considerably shorter than over more extensive laps, describing complex figures.

In the training stage, the Regularized Tuning Method finds a suitable parameter set in one of the small laps (clockwise and counterclockwise). The found parameters are tested in the more complex video streams, the *eight* lap, proving to be suitable to the unseen data input (generalization).

The results displayed in Figs. 15, 14, and 16 demonstrate the effectiveness of the tuning method. It shows that the parameter trained on the counterclockwise lap is well-suited for all areas of the environment, including those that were not utilized in the training phase. The “eight” lap is particularly significant because it entails changing directions and covering the environment completely, indicating a notable generalization capability for the method.

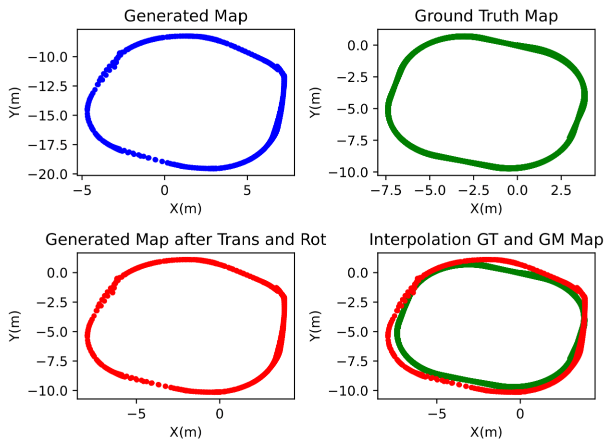
Finally, we used the proposed parameter tuning approach to enhance the parameters for *Lab Tour* environment (displayed in Tab. 2) using a *C++* implementation of RatSLAM (MUñOZ et al., 2022). These parameters were then minimally adjusted to fit the version of RatSLAM used in this work.

Figure 14 – EM of the counterclockwise lap.



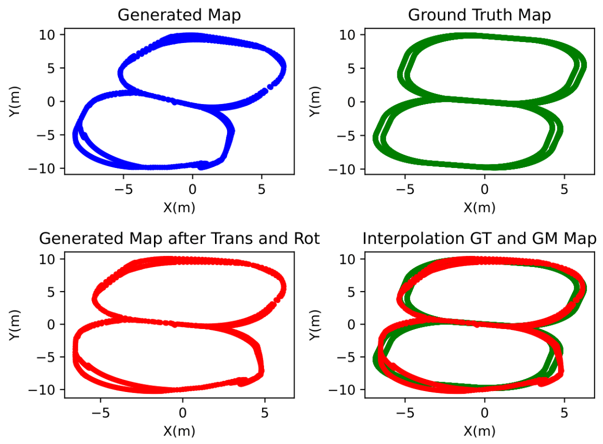
Source: Designed by the author and adapted from (GOMES et al., 2022)

Figure 15 – EM of the clockwise lap with counterclockwise parameters.



Source: Designed by the author and adapted from (GOMES et al., 2022)

Figure 16 – EM of the *eight* lap with counterclockwise parameters.



Source: Designed by the author and adapted from (GOMES et al., 2022)

4.3 Results

This section presents the results of the *Virtual Tour*, *Lab Tour*, *iRat*, and New College experiments. To compare the multisession solution with the standard RatSLAM process, the EMs for both multiple mapping and single mapping sessions are displayed.

4.3.1 *Virtual Tour* Results

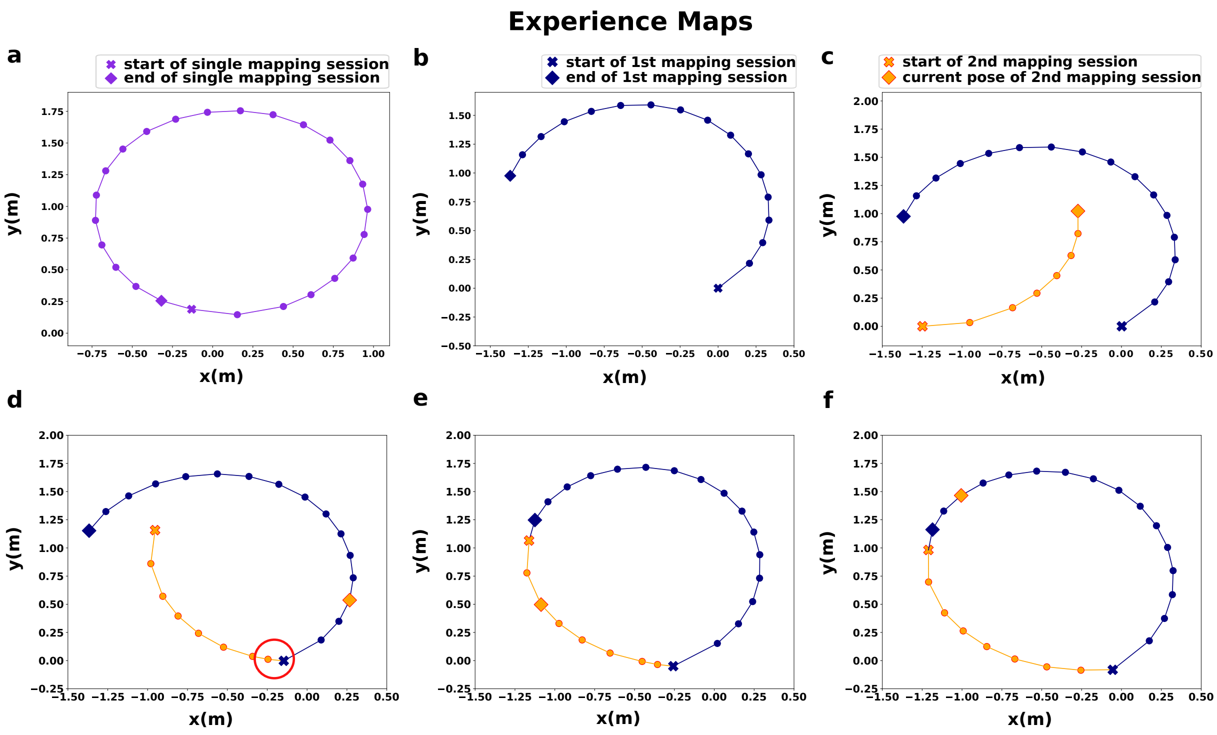
For comparison, the EM for the single RatSLAM mapping is displayed in Fig. 17a. The link between the start and endpoint of the EM shows that the loop is closed in the map. The result of the multisession mapping is presented in different stages to show the evolution of the EM, starting at the end of the first session (Fig. 17b). Figure 17c depicts the second session with the partial map (yellow), alongside the loaded map (blue), at the moment when the agent found an experience on its EM that matched with an experience in the loaded map, thus triggering the merge process. After the merge procedure, the experiences of the partial map were transformed (translation and rotation) and joined into the loaded map (Fig 17d). Once the agent completed a full lap in the second session, the EMs of the first and second sessions formed a closed loop (Fig. 17e). As expected, the number of EM nodes in the single and multiple sessions is the same, meaning no new experiences were created after merging the maps. Therefore, the merge operations made the creation of new experiences unnecessary. Neither new templates nor new activity packets on PCN were added to the EM.

Figure 17f displays the final merged EM of both the first and second mapping sessions. As displayed, a path correction is performed by RatSLAM over the EM. This correction on the merged EM shows that after operation iv), the nodes are linked in such a manner that they influence each other when the relaxation algorithm distributes the odometry error throughout the graph. Therefore, this path correction demonstrates that the RatSLAM works as expected after the merge procedure.

Finally, the ICP comparison of both single and multisession maps shows strong similarities between the paths after the transformation of the merged map to fit into the single-session map (Fig. 18). RMSE = 0.00773327.

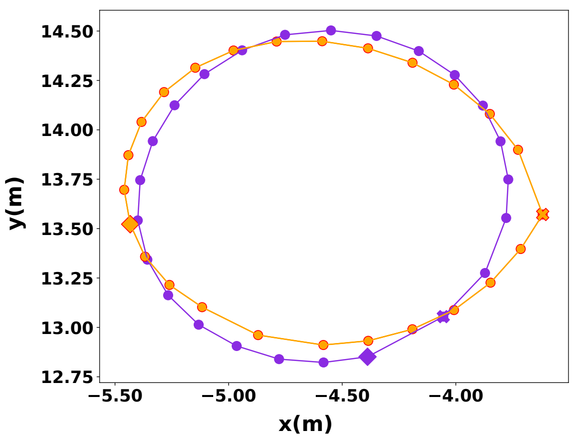
The behavior of the PCN is analyzed during the merge process. The Virtual Tour PCN was chosen because it has a simplified behavior due to the path mapped by the agent, i.e., the PCN is fully activated in the Θ axis (robot orientation) because of the ellipse-shaped laps. This behavior can be seen after the end of a single mapping session (Fig. 19a). It is worth mentioning that the second and third laps done by the virtual agent have the same frames and speed information as the first one. Hence, the PCN units from the first lap are reactivated in these next ones. Therefore, we define *start* and *end* for the first and last energy packets created in the single mapping session.

Figure 17 – Experience Maps (EM) of the Virtual Tour experiment.



Source: Designed by the author and adapted from (MENEZES et al., 2023)

Figure 18 – ICP comparison for *Virtual Tour* experiment.

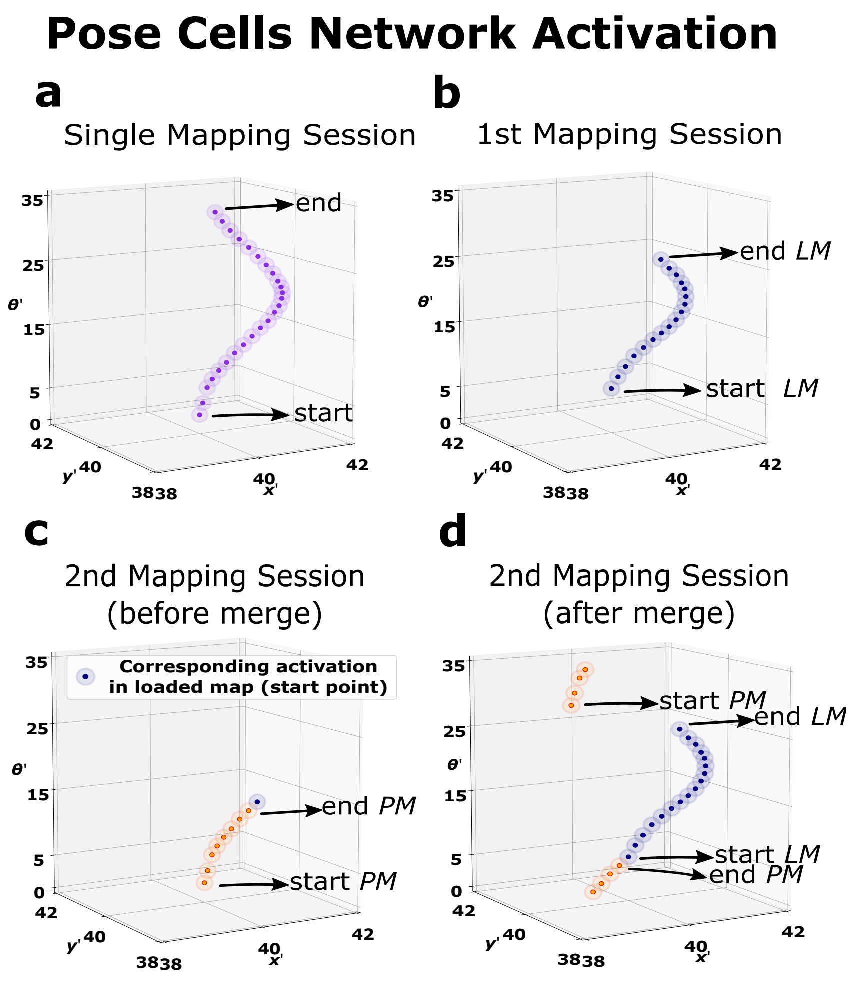


Source: Designed by the author and adapted from (MENEZES et al., 2023)

The PCN for the multisession is partially stored in the loaded and partial map structures. As the loaded map has not completed a full lap, its PCN is partially filled on the Θ axis (Fig. 19b). Likewise, the partial map also has only part of its PCN activated. Nevertheless, its last activation (blue) corresponds to the *start LM* activation of the loaded map (Fig. 19c). Through operation (iii) in the merge, the activities of the partial map are shifted to match the blue activity packet with the *start LM* in the loaded map. Note that part of the partial map PCN activity packets is shifted to the network's top face after reaching the bottom face's boundary.

Similar to the single SLAM session, the final merged map of the *Virtual Tour* is a closed-loop ellipse. Consistently, the final PCN is activated through the complete Θ axis after the merge (Fig. 19c). These results show that our method changes both EM and the PCN coherently, and their final results resemble the ones from a single mapping session.

Figure 19 – Pose Cells Network (PCN) activation in the Virtual Tour experiment.



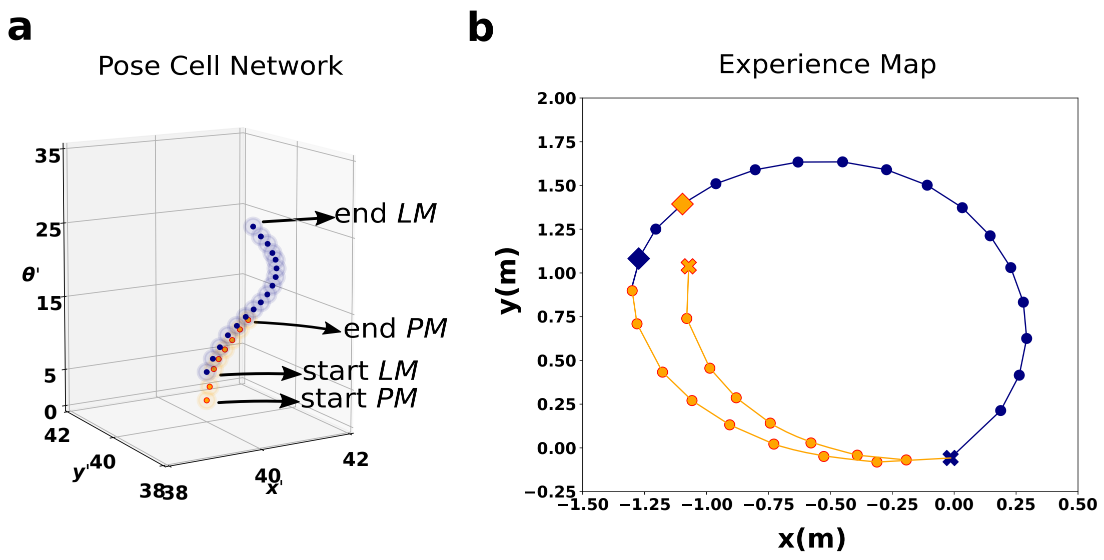
Source: Designed by the author and adapted from (MENEZES et al., 2023)

Finally, to demonstrate the influence and impact of the PCN operation on the merge process, multisession mapping was carried out without shifting the PCN activities of the partial map (operation iii). As expected, new experiences were created due to the incorrect association between templates and PCN activity (Fig. 20). Therefore, as demonstrated, the merge operations on the PCN are necessary for RatSLAM to perform correctly.

4.3.2 *Lab Tour* Results

Compared to the *Virtual Tour*, the *Lab Tour* experiment is more complex in two regards. It was performed in a physical environment using a robot in a more complex

Figure 20 – No shifting association between PCN and LVC.



Source: Designed by the author and adapted from (MENEZES et al., 2023)

environment's topology. Due to the robustness of RatSLAM, the EM for the single mapping session nevertheless correctly represents the environment's topology (Fig. 21a). So, does the multisession mapping, which we discuss step-by-step? After the first session, the EM only reflects one of the loops in the environment (Fig. 21b) since the agent only had experiences in the smaller loop. Up until the merge, conditions are met (Fig. 21c), the loaded and partial maps are not aligned, and their relative positions are random. This changes when the two maps are merged (Fig. 21d). Note that the nodes of the partial map had been concatenated at the merge point (red circle).

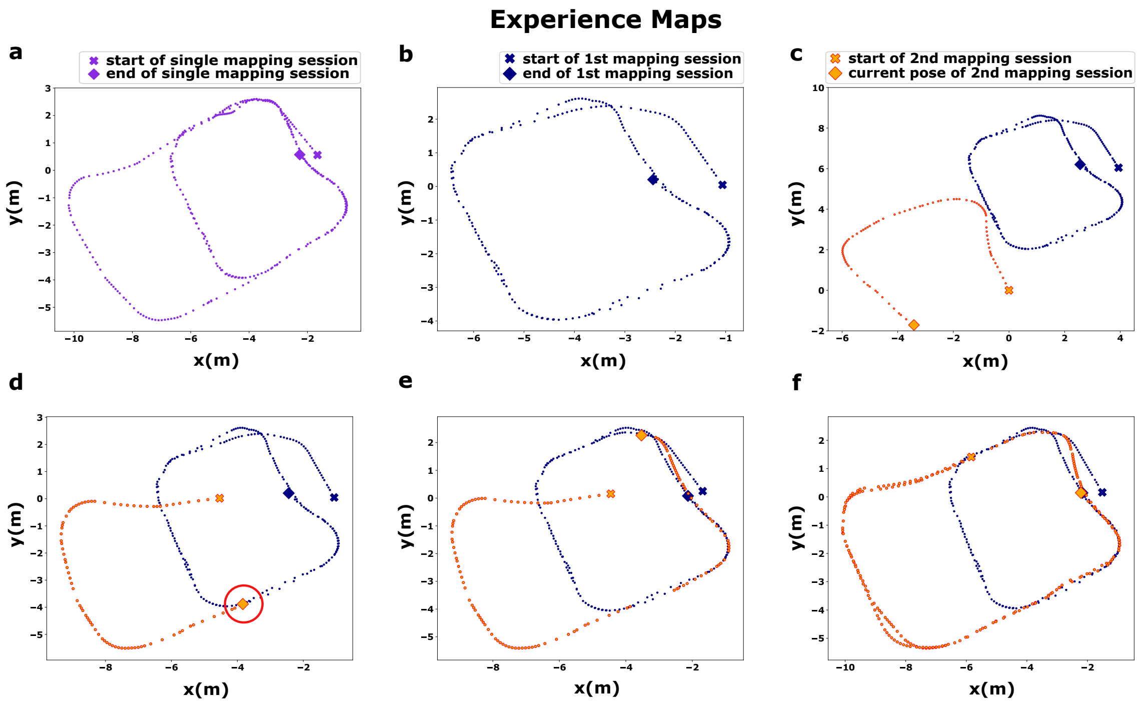
Figure 21e displays a path correction of the start and end nodes (blue cross and diamond symbols) of the first session map. Path correction over an already mapped place reinforces the expected execution of RatSLAM after the merge procedure since both maps are linked in one single structure. Thus, all the nodes should be affected if changes are needed in the EM after the merge. Besides that, this result shows that the merge approach could improve the mapping in multisession.

The final EM at the end of the second session exhibits the results of the second loop closure when the robot completes a full lap in the second session path Fig. 21f. This final map is similar to the EM generated in the single session (Fig. 22). RMSE = 0.00883269.

4.3.3 *iRat* Results

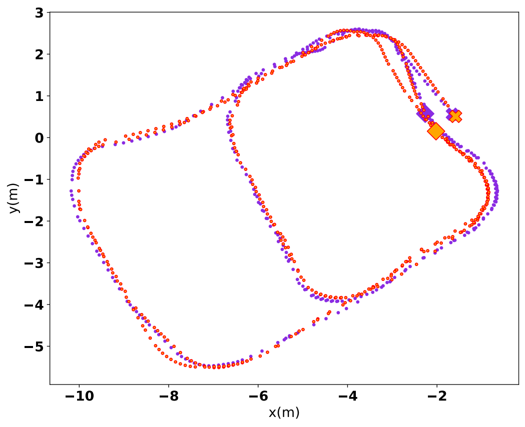
The *iRat* experiment is challenging since it requires merging maps in five sessions. Nevertheless, single-session RatSLAM performs very well, generating an accurate EM (Fig. 23a). Note that the robot continuously moves through the environment in a single

Figure 21 – Experience maps (EM) of the Lab Tour experiment.



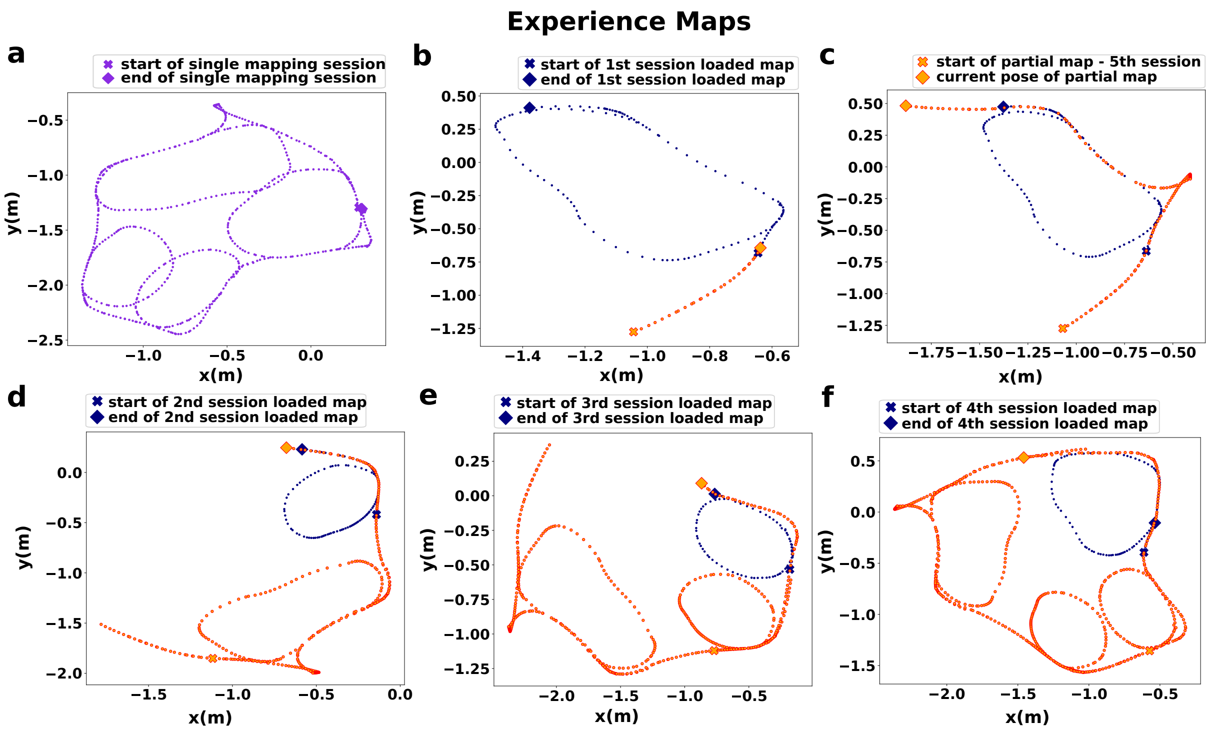
Source: Designed by the author and adapted from (MENEZES et al., 2023)

Figure 22 – ICP comparison for *Lab Tour* experiment.



Source: Designed by the author and adapted from (MENEZES et al., 2023)

Figure 23 – Experience maps (EM) of the *iRat* experiment.



Source: Designed by the author and adapted from (MENEZES et al., 2023)

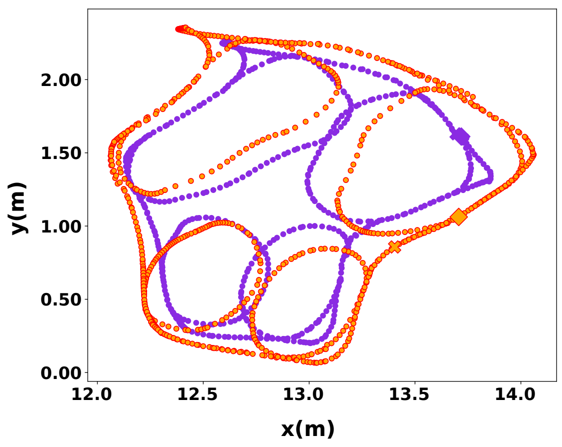
mapping session. This continuous movement influences the behavior of RatSLAM's relaxation algorithm (2.13) since the path correction depends on the connection among the nodes.

The results of the multisession mapping show that the fifth session seamlessly connected the loaded EMs of the previous sessions. (Fig. 23b-f). Unlike single-session mapping, multisession maps are not generated through continuous robot movements, that is, each session generates its map independently. In addition, the mapping of the fifth session covers only the external area of the environment. Therefore, the existing links between the loaded and partial maps are their regions corresponding to this external path. Without linking the internal nodes with the external ones, no path corrections can be made on the final map. This may have influenced the high distances shown in the internal maps compared to single-session EM (Fig. 24). Nevertheless, the ICP comparison between the final multisession EM with the single-session EM shows clear visual similarities between the internal paths and the overall map. (Fig. 24). RMSE = 0.0126932.

4.3.4 New College Results

The New College experiment is the biggest challenge since it involves mapping an even more complex environment, and the multisession version requires the merging of large maps. The single-session EM closely matches the physical structure of the environment (Fig. 25a). Similarly, the first and second sessions show similar maps for the

Figure 24 – ICP comparison of EM for *iRat* experiment.



Source: Designed by the author and adapted from (MENEZES et al., 2023)

different areas (Fig. 25b-c).

The partial map of the third session is quickly merged since it starts in a known place inside the first session loaded map (Fig. 25d, red circle). After completing its trajectory in the first session area, the resulting map is merged into the second session loaded map at its expected location (Fig. 25e, red circle).

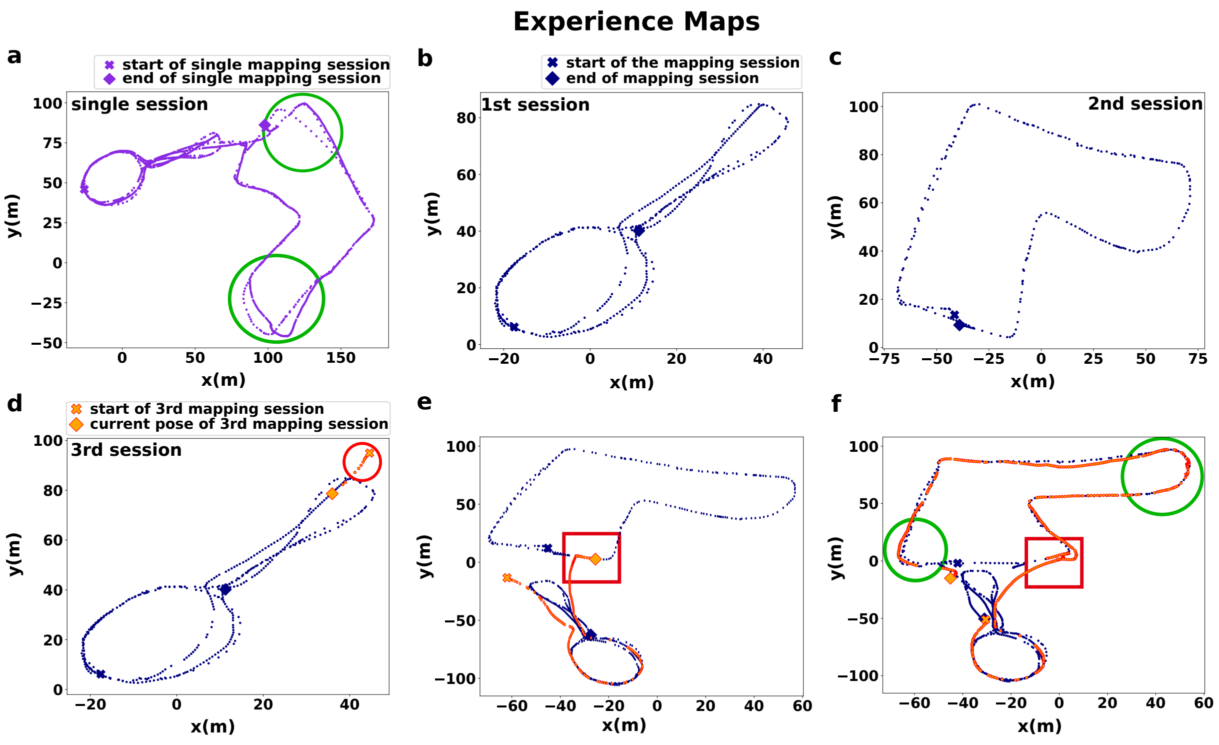
As can be observed, in continuing the mapping, the RatSLAM algorithm applied an inconsistent path correction at the second merge point, which affected the final map (red squares, Fig. 25e–f). This inconsistency led to a clear difference between single-session and multisession maps when compared by ICP (Fig. 26, RMSE = 85.4687). However, despite this inconsistent correction, the final multisession map presents a sufficient similarity of structures that compose the environment path (see Fig. 13). Notably, the multisession map correctly closed loops where the single-session mapping failed to perform it. (green circles, Fig. 25a,f).

4.4 Summary

This chapter introduced a new approach to the multisession SLAM problem for the RatSLAM algorithm. Our solution addresses the mapping scenario where the robot is not located in the previously generated map, and newly mapped areas have to be merged with previously mapped ones into a single coherent map of the environment.

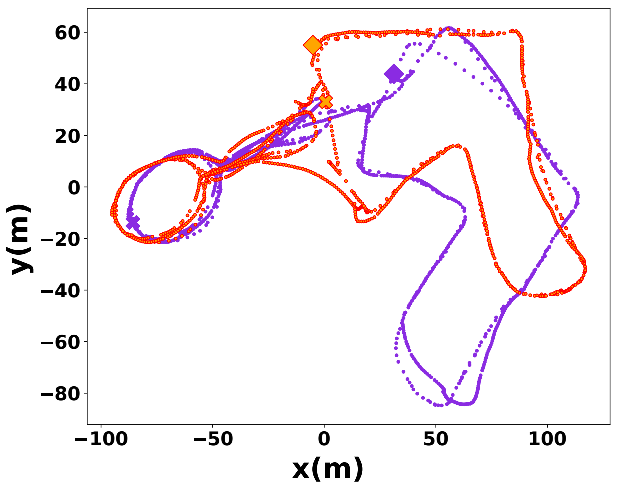
To merge RatSLAM maps, we employ a merge operation in its structures (LVC, PCN, and EM) acquired and stored during previous mapping sessions. Our approach involves merging the current map with previously stored ones when both maps share a template, adjusting all structures spatially. Once the merging is complete, the mapping session can continue, following the standard RatSLAM process and performing new

Figure 25 – Experience maps (EM) of the *Oxford New College* experiment.



Source: Designed by the author and adapted from (MENEZES et al., 2023)

Figure 26 – ICP comparison of EM for *Oxford New College* experiment.



Source: Designed by the author and adapted from (MENEZES et al., 2023) merging procedures as needed.

In this study, we examined four different environments: the *Virtual Tour* experiment, where a virtual robot followed an ellipse path; the *Lab Tour* experiment, which involved the mapping of a research laboratory using a robotic platform called RoboDeck; the *iRat* dataset, which was used to validate the openRatSLAM implementation; and the challenging *New College* dataset. For the *Virtual Tour* and *Lab Tour* experiments, two mapping sessions were conducted, while the *New College* and *iRat* experiments were divided into three and five mapping sessions, respectively

The results of the experiments indicated the following: a) EMs successfully performed path corrections; b) no new experiences were generated during the *Virtual Tour* experiment; c) the PCN activity packages for *Virtual Tour* in a single mapping session were similar to those of the multisession; d) EM matching was effective in both the real-world *Lab Tour* and the realistic *iRat* experiments; e) loop closure was accurately performed compared to single-session mapping in the *New College* dataset. Overall, this study demonstrates that the proposed merge mechanism effectively addresses multisession solutions (i) and (ii) and can serve as a solution for the RatSLAM algorithm.

5 Spatial Learning Based on Cognitive Maps

This chapter presents our study of two important processes in spatial navigation, mapping and learning. For this, we modify CoBeL-RL, adding RatSLAM to its learning process as a State Representation Learning (SRL). Therefore, RatSLAM dynamically acquires and updates 2D spatial states from unknown environments to CoBeL-RL.

Initially, we present how the RatSLAM was integrated into the CoBeL-RL process, along with simulations demonstrating the framework's usage in rewarded spatial tasks. By adding RatSLAM, we introduced new variables that directly impact the physical exploration of the environment and the process of learning rewards.

Subsequently, we employ this framework with the Dyna-DSR agent already implemented in CoBeL-RL to investigate latent learning computationally. We use this computational approach to comprehend experimental results on learning performance observed in animals during latent learning over various experimental designs and exploration strategies.

Specifically, we examine the contrasting effects of the type-one and type-two experiment designs (KARN; JR., 1946) and delve into the exploration strategies involved in learning cognitive maps, including changing the RL agent policy and implementing maze structures with ports (DAUB, 1933).

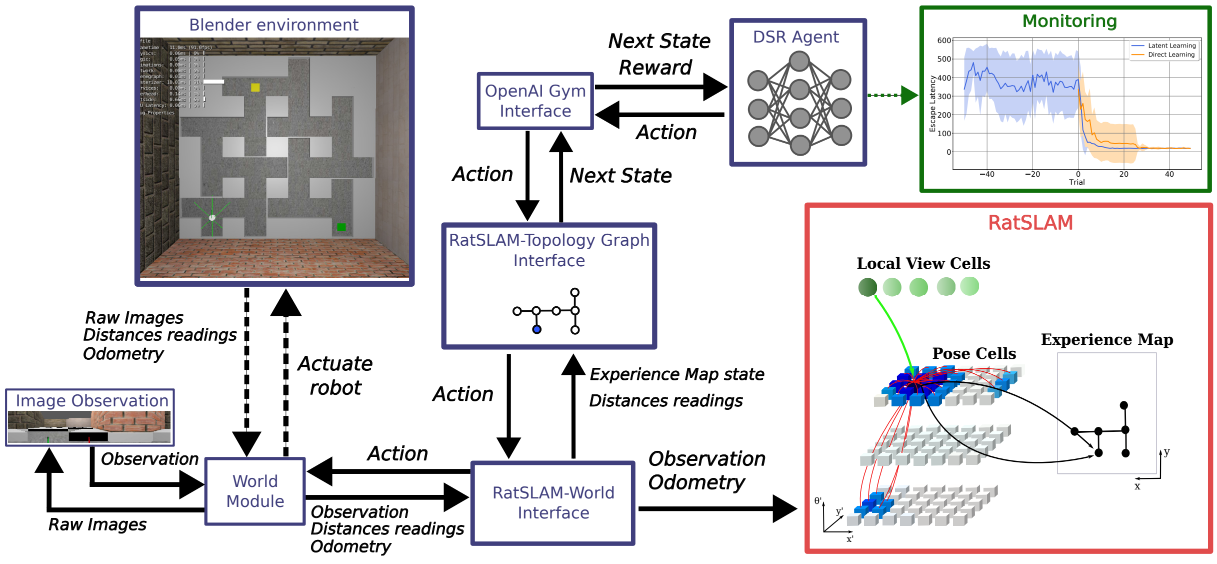
5.1 CoBeL-RL and RatSLAM Integration

The neural-inspired approach for mapping and learning in unfamiliar environments builds upon the existing functionalities of CoBeL-RL (Fig. 27). The primary function of RatSLAM is to construct the Experience Map (EM), which serves as the fundamental component for the Topology Graph used by CoBeL-RL. Whenever the CoBeL-RL agent acts in the environment, RatSLAM updates the Experience Map based on changes in the environment's observations. Interface modules are implemented to facilitate communication between RatSLAM and other CoBeL-RL modules.

Our approach also processes information from the virtual robot's distance sensors in the 3D environment modeled in Blender Game Engine (BGE) to enable autonomous navigation. We built a set of proper actions from these data to avoid collisions with environmental obstacles. While this additional information is optional for the functioning of CoBeL-RL, it plays a vital role in decision-making when mapping unknown environments.

The following paragraphs describe the framework's functioning process.

Figure 27 – Integration of RatSLAM in CoBeL-RL framework.



Source: Designed by the author

The World Module connects with the BGE through web sockets (dashed lines) (DIEKMANN et al., 2023). This module supplies raw images, distance sensors, and odometry data from the virtual agent to the CoBeL-RL. The distance sensors return measurements from four directions, enabling the framework to gather insights about the robot's surroundings. Odometry informs about the linear and angular velocities of the virtual robot. Raw images from the robot's four cameras are processed in the Image Observation step to create a panoramic view of the surroundings. These panoramic frames serve as *Oservations* for CoBeL-RL, with dimensions set at 256x64 pixels. For RatSLAM, we use grayscale when working with these observations.

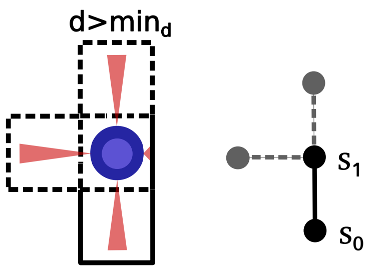
The RatSLAM-World interface module interfaces RatSLAM with CoBeL-RL. From this module, a RatSLAM interaction can be called, receiving observation and odometry as inputs, resulting in an internal change in RatSLAM's structures. For example, Local View Cells (LVC) templates and Pose Cells Network (PCN) activity may be updated. Ultimately, the EM is updated based on the PCN and LVC changes. We rely on EM updates to maintain a mutable Topological Graph within CoBeL-RL.

The RatSLAM-Topology Graph interface employs processes that convert the EM information into a CoBeL-RL Topology Graph structure. Since the Topology Graph is a standard component in CoBeL-RL's spatial representation class (DIEKMANN et al., 2023), the proposed interface integrates the EM data without making critical modifications to the CoBeL-RL structure. The conversion from EM to the topological graph can be performed at various levels. The most straightforward approach involves directly mapping the information from EM experiences onto the graph. Alternatively, for example, one can filter out specific experiences to build a simpler map. In this work, we directly mapped the Topology Graph from the EM, with each experience in the EM corresponding to a node in the Topology Graph.

The Topology Graph incorporates structured information about neighboring nodes based on the actions required to reach them. In our setup, four actions can be performed by the agent: right, up, left, and down. Figure 28 provides an example of state s_1 neighbors, represented by the array $s_{1\_\text{neighbors}}$. The actions that are known or have been performed from s_1 are depicted in n_{action}, which shows the action "down" and refers to reaching the node s_0 (solid line).

Additionally, the structure of the Topology Graph has been modified to include what we term "non-engaged actions" or "possible actions". These actions represent areas within the environment where the robot can physically explore but has not yet engaged. To automatically determine these possible actions, we leverage information from distance sensors. Figure 28 depicts p$_{\text{actions}}$, which represents the possible actions of state s_1, determined by the sensor reading d that exceeds the threshold min$_d$. Engaging in one of these possible actions may create a new experience in the EM and generate a new node on the map.

Figure 28 – Possible actions.



$s_{1\_\text{neighboors}}$ = [-1, -1, S_0, -1]
n_{actions} = [down]
p_{actions} = [up, left]

Source: Designed by the author

By differentiating between known and potential neighbors, our framework can dynamically choose between continuing to learn within the known states or acquiring new information by exploring new parts of the environment induced by curiosity (PATHAK et al., 2017). This flexibility allows for studies on the trade-off and how it may influence the rate at which the spatial task is learned.

OpenAI Gym Interface calls a "step" within an action from the agent policy decision. Engaging in this step means the agent will act in the environment. For our model, this accounts for the action being sent through the aforementioned modules until the virtual robot actuates over the environment. This irradiates changes in the robot's observation, odometry, and distance readings, ultimately changing the current state of the environment in the Topology graph. This change is so defined as the next state for the RL agent. Additionally, the *OpenAI Gym* module defines the reward for each state.

Together with the next state, the experience tuple (s_t, a_t, r_t, s_{t+1}) is sent to the Dyna-DSR agent. The Dyna-DSR (see Chapter 2) agent keeps the information of the current state. The agent implements a small DNN to approximate the SR for each action, which we refer to as *deep SR*. The input for the DNN is an *Observation*, which consists of a one-hot encoding vector for each state based on the node index of the spatial representation. Besides that, the agent uses a tabular $n \times 4 \times n$ memory to store the experiences and perform replay, where n is the number of states.

The final component of the RL agent module involves action selection, where actions are determined based on the RL agent's policy or a mechanism for physically mapping the environment. This denotes the agent can select from actions that lead to known states or engage in a possible action and learn a new state. We model this dynamic action selection process by a random variable p_τ under a constant value τ. If $p_\tau < \tau$, an action is selected from p_{action}; otherwise, it is chosen from n_{action}. In cases where multiple possible actions exist, the selection of p_{action} is randomized.

Furthermore, we monitor the agent's performance and ability to obtain rewards through the CoBeL-RL utility module monitors. These monitors include escape latency, reward monitoring, and more, allowing us to assess the agent's behavior.

5.2 CoBeL-RL and RatSLAM in Rewarding Spatial Task

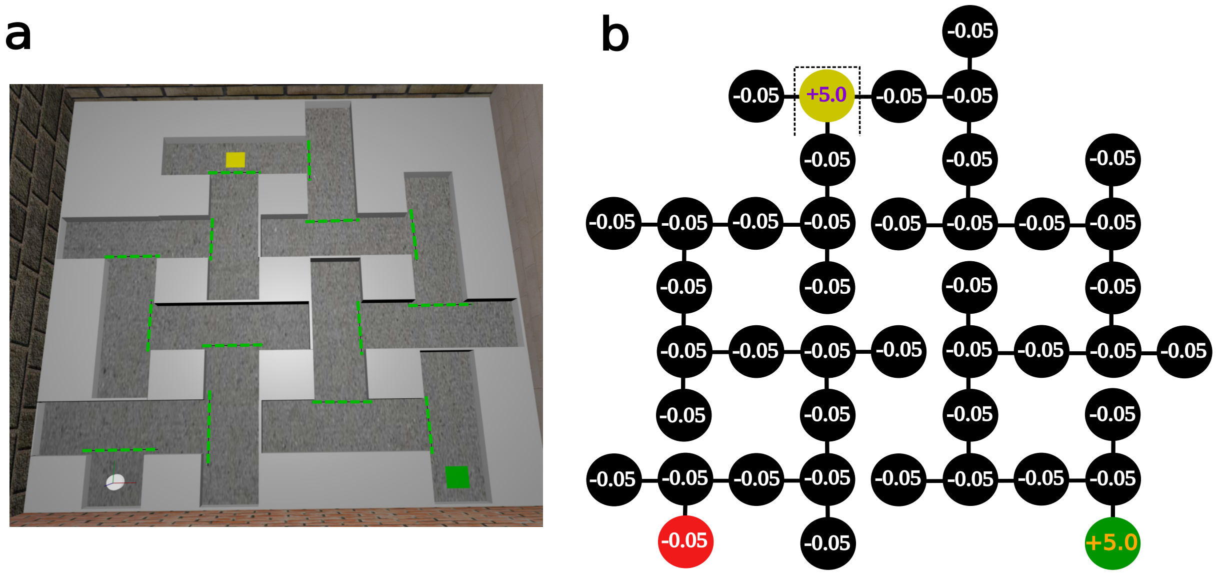
In this first experiment, we evaluated the framework's performance in learning to locate a reward within unknown environments operating in either Blodgett or Tolman's setup.

Experimental Setup

To simulate the task, we constructed a 14-T virtual maze using Blender (Fig. 29a). The goal location for our experiments was set to either the yellow or green squares. The yellow square corresponds to a maze configuration resembling the 6-arm Blodgett T-maze, while the green square represents the 14-arm Tolman T-maze (TOLMAN, 1948). Physical walls, i.e., solid blocks, were incorporated to maintain the integrity of the maze, preventing the agent from executing infeasible actions such as those traversing through the walls. Additionally, we incorporate invisible virtual doors (dashed green lines) that can be used if enabled. We use these doors in future simulations.

We conducted simulations with agents using different true-exploration parameters

Figure 29 – Virtual Environment.



Source: Designed by the author

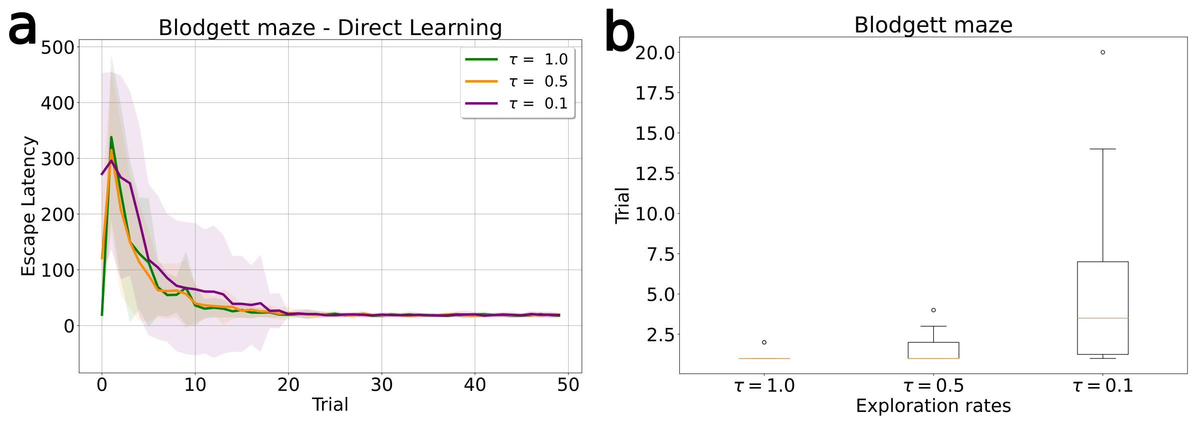
for each maze configuration. Specifically, we used $\tau = [0.1, 0.5, 1.0]$, totaling 3 different agents, for the Blodgett maze and $\tau = [0.1, 0.5, 0.8, 1.0]$ for the Tolman maze (4 different agents). During each step, the agent receives a minor punishment of (-0.05) as part of the exploration strategy. This penalty incentivizes the agent to further explore the environment under the RL paradigm (different from the true-exploration strategy), as the policy aims to maximize the long-term reward. In our setup, depending on the experiment, when the agent successfully reaches one of the goals in either the Blodgett or Tolman locations, it receives a reward of $+5.0$. We monitor the agent's Escape Latency to reach the goal states as a behavioral variable for each experimental setup, with 30 simulations per agent. The agents learned an ε-greedy policy with $\varepsilon = 0.3$ and a replay batch size of 32 randomly recovered experiences. In addition to the Escape Latency monitoring for performance analysis, we captured the trial in which the agent learned all states for the environment. This additional information allows us to discuss the relationship between the true-exploration performance and p_τ value.

Moreover, we also captured in which trial the agent learned the total number of states in the environment.

Results

Blodgett's maze experiment results show that the value of τ affects the performance at which the agent learns the rewarding task. Agents with $\tau = 0.1$ exhibited slower learning performance (Fig. 30a), as indicated by the higher standard deviation observed in each trial. Conversely, agents with $\tau = 0.5$ and 1.0 displayed similar learning performance. The variation in τ also impacts the speed at which the agent captured (learned) the complete environment states. The average trial at which the agents acquired full knowledge of the environment state revealed that lower τ values resulted in slower learning (Fig. 30b).

Figure 30 – Reward learning in Blodgett's maze.



Source: Designed by the author

In the Tolman setup experiment, which is more complex regarding the number of states and path length to reach the rewarded state, the τ value differences significantly impact the agent performance. For $\tau = 1.0$, agents learn optimally the task around the 15th trial (Fig. 31b). However, escape latency became stable but did not present an optimal number of steps to reach the goal, for agents with $\tau = 0.8$, $\tau = 0.5$, and 0.1. Therefore, the decrease of τ values follows the increase of steps (on average) to complete the task.

Similar to the findings in the Blodgett experiment, varying τ in the Tolman setup also influences the speed at which the agent learns the complete environment states. The average trial at which the agents learned all the environment states revealed that lower τ values resulted in slower learning (Fig. 31b). Outliers show that agents operating at $\tau = 0.5$ and $\tau = 0.1$ failed to acquire the entire set of states.

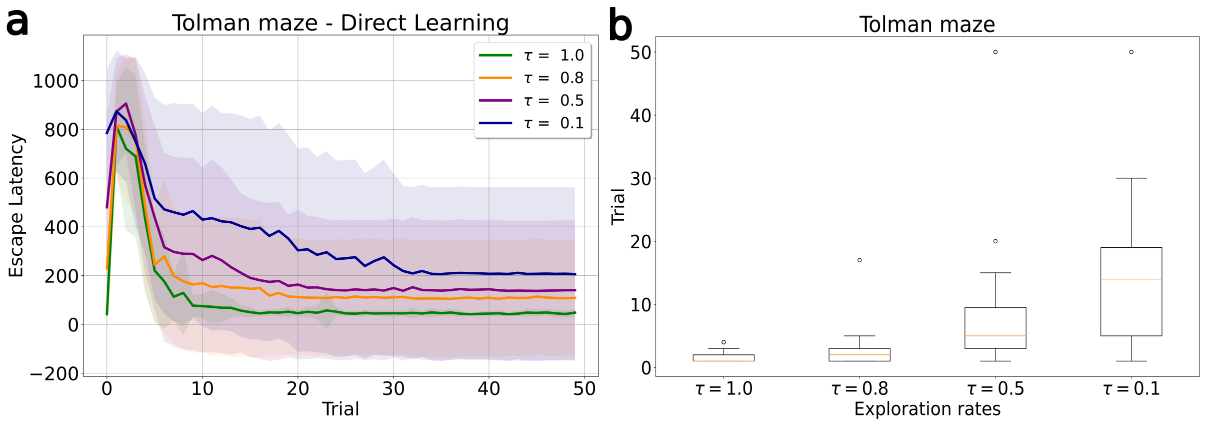
5.3 Latent Learning During Spatial Exploration

In our next investigation, we employ our proposed framework in learning reward tasks through latent learning. We train two types of agents, direct learning and latent learning, using Blodgett and Tolman's maze setups.

Experimental Setup

The experiment setup for latent learning follows procedures similar to those adopted by Tolman and Blodgett (TOLMAN, 1948). Like the rewarded control group in Tolman's experiment, direct learning agents receive rewards as they reach the goal location from the first trial. On the other hand, latent learning agents receive the reward only after a specific number of trials, which consists of 50 trials of pre-exposure in the

Figure 31 – Reward learning in Tolman's maze.



Source: Designed by the author

empty and 50 learning trials with a reward. Both types of agents follow an ε-greedy policy with $\varepsilon = 0.3$. For the Blodgett maze, agents have a maximum of 500 steps, while the Tolman maze allows a maximum of 1000. The latent learning is single-trial (ST) agents, meaning the simulation stops when they reach the goal location or have performed the maximum number of steps. During each trial, we capture the agents' Escape Latency and monitor if they reach the goal location in each trial. For each agent, 30 simulations were performed.

We modify the way we acquire the environment states in comparison to the previous experiment, aiming to standardize the analysis of the agents' successor representation (SR) and Q-values. Instead of mapping states based on the value of τ, we map the states in the same predefined order sequence before the agents start to learn. Nevertheless, we still utilize RatSLAM to acquire and inform CoBeL-RL about the agent's current state. This approach ensures that the representation of states in the topology graph is standardized and consistent. The mapping approach resembles $\tau = 1.0$, which prioritizes unexplored locations in the same order of mapping. Moreover, the reward associated with each state is not computed during the acquisition of states in the predefined mapping.

Results

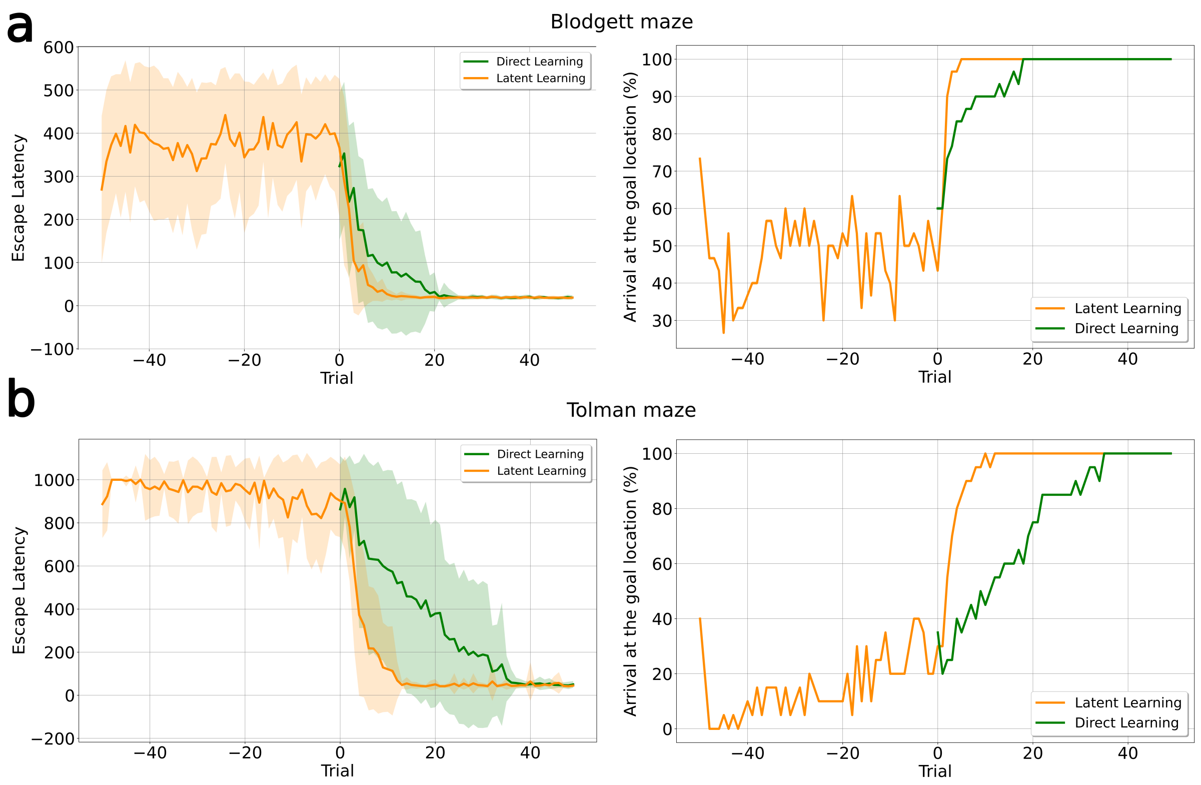
Latent learning agents learn the rewarding task faster than direct learning agents after the pre-exposure phase in both environments (Fig. 32). This latent learning behavior is observed as a significant drop in Escape Latency curves as they experience reward in the environment. Escape Latency curves were shifted to simplify behavior analysis during unrewarded and rewarded trials. Thus, for both agents, the rewarded trials start at 0.

Furthermore, during the unrewarded phase of the experiments, the Escape

Latency of latent learning agents indicates no apparent learning toward the goal location in both mazes (trials -50 to -1). Nevertheless, in the rewarded phase, these agents stabilize their learning curves near trial 10, showing consistent performance in the Blodgett maze. In contrast, direct agents reach a similar stabilization point around trial 20. Notably, both agents present a more pronounced disparity in the Tolman maze. The latent agents were able to stabilize their escape time to reach the goal by trial 15, whereas the direct agents took around 40 trials to do the same. The results align with the behavioral findings in Tolman's and Blodgett's latent animal learning experiments (TOLMAN, 1948).

Latent learning agents' success rate of reaching the goal location (Fig. 32a,b, right panels) does not exhibit significant improvement as during the pre-exposure period. This absence of improvement suggests that no apparent learning occurs during the unrewarded learning phase.

Figure 32 – Agent's escape latency in Latent Learning.



Source: Designed by the author

The agents have learned a deep SR structure corresponding to the optimal path leading to the goal location in both environments (Fig.33). Each row of the deep SR matrix represents the expected future visits to other states starting from the state with the same index. For instance, the deep SR row 0 (first row) shows the SR of state $= 0$. Moreover, as the Dyna-DSR learns a deep SR matrix for each action, we extracted each row from the optimal action for that state to build a single deep SR matrix. For example, the first row of the deep SR representing state $= 0$ is from the deep SR with action “up”, while the SR of state $= 1$ was extracted from the deep SR with action “right”.

The lighter states in Fig.33a illustrate the optimal sequence of states followed in Blodgett and Tolman’s mazes, in which states 19 and 42 correspond to the goal states in each respective maze. After completing the rewarded task, a comparison of the deep SR matrices of the direct and latent learning agents (Fig. 33b,c, comparison between trials 49, right panels) demonstrates their similarity. Trials exhibited in deep SR results are equivalent to those shown in the Escape Latency in Fig. 32. Thus, trial 0 means the beginning of the rewarded phase, while trial -50 refers to the init of the pre-exposure period. It is worth noting that as the state approximates to goal one, their SR values tend to increase to forward states, showing that backward states are not expected to be visited in the future. A noteworthy observation is that as the state approaches the goal, the SR values have an increasing tendency towards the states ahead. This indicates that past visited states are not anticipated to be visited.

Direct learning agents deep SR shows progressive learning towards the optimal path representation over trials in both the Blodgett maze (Fig.33b, top row) and the Tolman maze (Fig.33c, top row). The deep SR values from trials 0 result from the agent’s deep SR network layers initialization, which follows the Glorot uniform algorithm (GLOROT; BENGIO, 2010). As the learning process continues, the deep SR matrix shows that each state is anticipated to visit the states that lead to the goal. This indicates that the direct learning agents successfully acquire a representation of the optimal path in both maze environments.

After the pre-exposure period, latent learning agents demonstrate evidence of having acquired a deep SR that differs from the network initialization (Fig. 33b-c, trial 0, bottom row). This learned structure indicates that each state has developed an expectation of visiting its neighboring states. Moreover, a direct comparison between the deep SR matrices in Tolman’s maze on trial 24 reveals a noticeable difference, with the latent learning structure being more similar to that observed at the end of the task (trials 49). This can be explained by the fact that by trial 24, latent learning agents had achieved stable performance in the tasks.

Individual analysis of deep SR states in latent learning agents provides evidence that the representation acquired by these agents at the end of pre-exposure is characterized by a high frequency of expected visits to their neighboring states (Fig. 34). In trial -50, deep SR values from states 0 and 10 in the Blodgett maze and states 0 and 19 in the Tolman maze are randomly distributed to the other states according to the initialization of the network (Fig. 34a-b, left column). The states’ positions in the mazes are marked with yellow crosses, and the dashed line represents the optimal trajectory from that state to the goal. The initial and goal locations are identified with red and blue borders. At trial 0, deep

SR values are higher for closer states and lower for more distant ones. At finishing simulations with a reward in trial 49 (right column), the deep SR values indicate discounted visits to states along the optimal trajectory.

Analysis of the Q-values reveals that a learned policy emerges only during the reward phase (Fig 35). The intensity of the graph's color symbolizes the values associated with each state-action pair, normalized between $[0, 1]$. In the 0 test, the Q-values of both agents in the mazes show no observable structure toward reward, which indicates that there is no significant policy learning during the pre-exposure phase for latent learning agents. On the other hand, the Q-values became distinct for the optimal action during and after the reward learning phase. Exceptions are seen in Tolman's maze, where latent learning agents show equally high Q-values for all actions in states near the end of the maze. We discuss the impact of these relative values in later sections.

5.4 Impact of Subtle Differences in Task Design on Latent Learning

The previous simulations have shown the efficacy of our computational approach in learning within an unknown environment. Using the Dyna-DSR agent, our approach also presented the ability to perform latent learning over the pre-exposure phase in a reward-devoid environment.

In this next section, we employ this proposed approach to investigate the impact of subtle different task designs on the acceleration of the learning process in latent learning. Specifically, we build upon the experimental findings of Karn e Jr. (1946), in which was observed different learning performances in two different experiment designs, namely type one and type two (THISTLETHWAITE, 1951).

Experimental setup

Type one experiment involves a single run in which the animal explores the environment. Once the animal reaches the desired location, it is removed from the maze. This type of experiment design corresponds to the Single-Trial (ST) utilized in previous simulations in this study.

In type two experiments, animals can live or explore the maze for a specific time. Our simulations refer to this as Time-Fixed (TF) exploration. Unlike ST experiments, animals are not removed upon reaching the goal location. Therefore, the goal location is not a terminal state for the agent, and a trial only ends upon reaching the maximum number of steps. While ST experiments always start at a specific state, state 0, TF simulations initialize the agent from different pre-defined states. In the Blodgett maze, these states are $[0, 8, 14, 19]$, with the last being the goal state in the reward phase. In the

Tolman maze, the pre-defined starting states are $[0, 8, 14, 19, 23, 29, 36, 42]$, with the last being the goal state. At the reward learning phase, all initiate from state 0, regardless of the design experiment.

We compare the TF simulation results with the ones presented in the last latent learning experiment with ST agents. Equivalent to ST agents, the TF agents learn under the ε-greedy policy with $\varepsilon = 0.3$ and the replay batch size of 32.

Results

Agents under TF exploration demonstrate latent learning compared to direct learning in the Blodgett and Tolman maze setups (Fig. 36a,b). Regardless, the Escape Latency reveals a difference in learning performance between ST and TF, with ST agents showing faster error drops and stabilization in their escape latency. These results qualitatively reproduce the results from (KARN; JR., 1946). It also aligns with the simulations from (SCLEIDOROVICH et al., 2020), where similar qualitative results indicate an advantage for the ST design experiment in terms of learning speed.

We hypothesize that, as the ST exploration is similar to the rewarded task in terms of how both tasks are conducted – both end when the agent arrives at the destination location – this causes agents of this design to learn a representation of the environment during pre-exposure similar to those of the rewarded task. On the other hand, TF experiments do not associate the goal location with the end of exploration, resulting in a representation of space less similar to the ones of the rewarded task. We investigated the underlying mechanism of these velocity differences by looking at the deep SRs of the ST and TF experiment designs.

To quantify the similarity between the deep SR matrices of the ST and TF designs and the rewarded task (referred to as the target SR), we employ cosine similarity (Fig. 37). This involves comparing each row of the ST and TF deep SR matrices with the corresponding row in the target SR matrix. Blodgett and Tolman's setup are presented in Fig. 37a and Fig. 37b, respectively. The cosine similarity scores reveal the level of similarity between the target SR and the final states of the ST experiments. The TF deep SR also exhibits some similarity, although it is relatively lower than the ST design. These results suggest that the final states of the deep SR under the ST design, even without reward-based learning, resemble the deep SR obtained from direct learning agents trained with rewards.

In addition, we examine the Q-values that emerged from the agents' deep SR learned by the end of the pre-exposure phase. We introduce a reward at the goal location to compute the Q-values from the agents of ST and TF designs and compute the inner product with their deep SR (Fig. 38). By analyzing the policy that has been learned, we can gain insight into the action-selection decision that expedites learning for ST agents.

Furthermore, from the Q-values, we computed the actions with higher Q-values in each state (Fig. 39). ST design exploration shows the longest optimal action sequence associated with the optimal path compared to TF design in both mazes. Circles display the state where the sequence begins. By starting from this state and following a greedy policy, ST-based agents would correctly reach the goal state for nine sequences of correct actions against six for TF in the Blodgett maze. In Tolman's maze, these sequences are 14 for ST against 10 for TF.

By comparing the SR and the actions generated by the design tasks, the presented results show that the ST experiment learns a representation of the environment closely associated with the rewarded task.

5.5 Impact of Exploration on Latent Learning

In this section, we will examine how exploration affects latent learning. One area of interest is the operation of doors, which restricts agents from revisiting certain areas. Additionally, we investigate different policies that the Dyna-DSR agent can adopt as exploration-exploitation strategies. Investigating latent learning through different policy analyses is crucial to evaluate performance. Moreover, comparing policies on SR-based agents can help us comprehend their characteristics on learning performance after the pre-exposure period. By evaluating policy performance, we can identify areas for improvement and refine existing policies, leading to enhanced performance and more effective latent learning.

Experimental setup

In the latent learning experiment with door, direct and latent learning agents adopt the ST design. Our experiment design differs from the approach taken by (DAUB, 1933) as the doors remain present during pre-exposure and reward tasks. In contrast, Daub (1933) only allowed the doors in the environment during rewarded training. We adopted the ε-greedy policy with $\varepsilon = 0.3$ and a replay batch of size $= 32$.

Under the ε-greedy policy, presented simulations showed that latent learning agents find it difficult to locate the goal in the initial attempts (Fig. 32b). This difficulty, which impacted the delay in learning the task, is attributed to how the agent must explore the environment. We compare the ε-greedy with *softmax policy* (SUTTON; BARTO, 2018) as an alternative to exploration, improving the agents to find the goal location more frequently in earlier trials.

An advantage of the softmax policy is that it selects the action based on a probability function based on the Q-values. However, if these values are similar, the selected actions can become random (depending on the temperature parameter used in the function). This problem does not apply to ε-greedy, which selects actions that maximize the Q-values within a certain probability. Regardless, in its exploration step, ε-greedy selects random actions that can not be suitable for space exploration. In space navigation, an exploration strategy to be adopted could be based on exploring states that were less visited.

In this context, we developed a new policy called "inhibition-greedy" that integrates the ε-greedy policies with *softmax*. The inhibition-greedy policy depends on the value of a random variable value $e \in [0,1]$ and selects a greedy action selection if $e < \varepsilon$. Otherwise, it engages in an action selection in a softmax function. However, the softmax does not compute the probabilities over the Q-values. Instead, it computes the probabilities over the recent state visitation to inhibit the selection of actions that lead to recently visited states. This is modeled in an inhibition matrix $I_{n_s \times n_a}$, in which n_s is the number of states and n_a is the actions state space equals to 4. The inhibition-greedy policy selects the action as follows:

$$\pi(s,\varepsilon) = \begin{cases} \frac{e^{\beta \cdot I(s,a)}}{\sum_k e^{\beta \cdot I(s,a_k)}} & \text{if } e \leq \varepsilon \\ \\ \text{argmax}_{a \in \mathbf{A}(s)} Q(s,a) & \text{otherwise} \end{cases} \tag{5.1}$$

Each time a state-action pair (s,a) is selected, the inhibition matrix I is updated with a maximum negative value of -5. Then, I is updated to decrease the inhibition value by a discount value γ:

$$I(s,a) \leftarrow \gamma I(s,a), \forall s \in S, \forall a \in A \tag{5.2}$$

In the experiments that adopt the inhibition-greedy policy, the $\gamma = 0.98$ and $\varepsilon = 0.3$ for both inhibition-greedy and ε-greedy policies. The inverse temperature in *sotmax* policy is $\beta = 5$ in the Blodgett setup and $\beta = 10$ in the Tolman maze.

Results

Using doors in our setup shows latent learning (Fig. 40). Latent learning agents display a decrease in escape latency during both rewards after pre-exposure periods. Similarly to latent learning experiments without doors, escape latency errors depict no apparent learning during the pre-exposure phase. Furthermore, the use of doors has been found to improve the learning process for both direct and latent agents. In comparison with latent learning without doors (see Fig. 32b), direct learning agents in Tolman's setup show a faster error decrease.

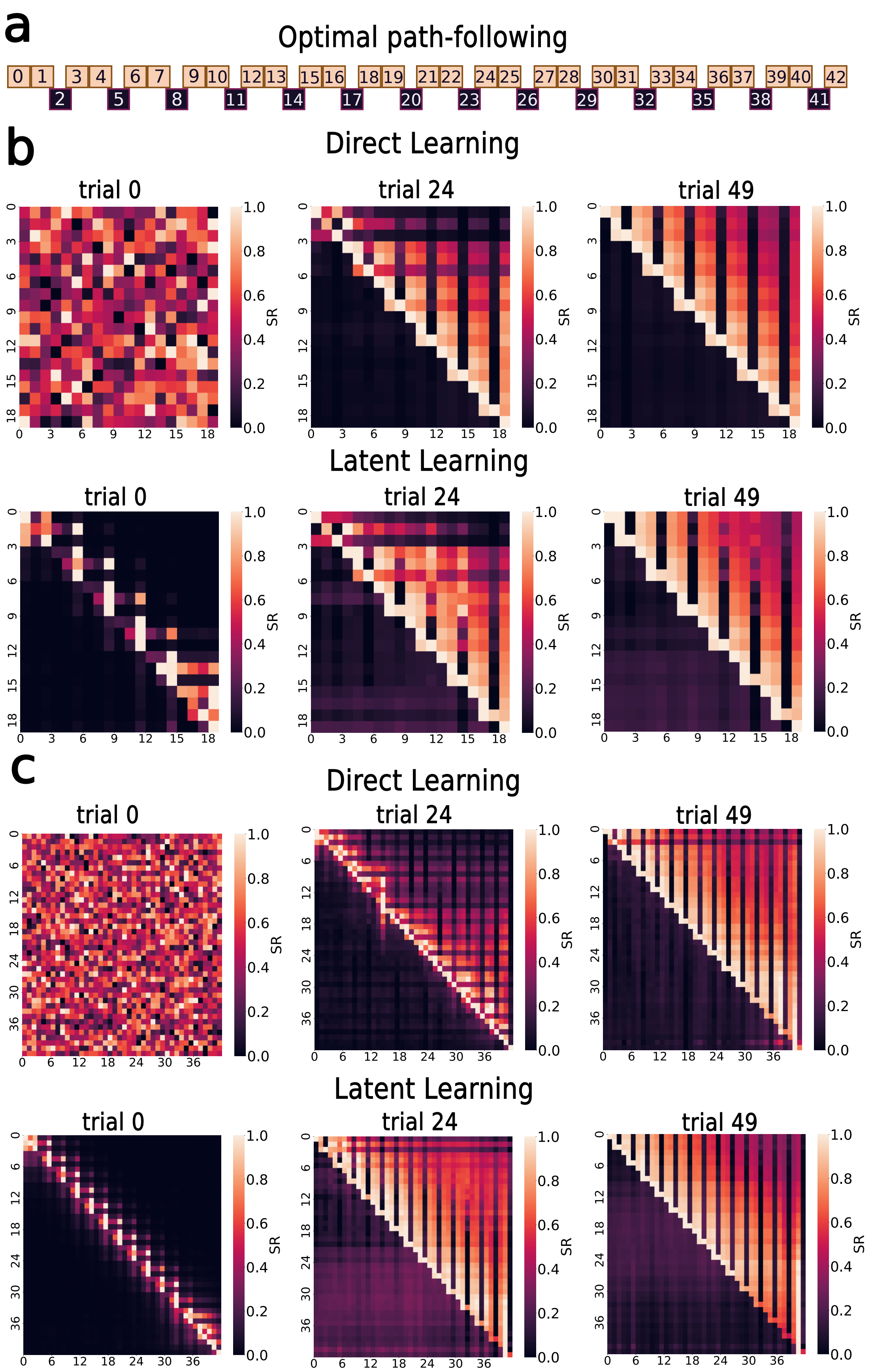
Latent learning is observed in both environments under different policies (Fig. 41a,b). When considering the escape latency during the non-rewarded phase, the softmax and inhibition-greedy policies exhibit similar or superior performance compared to the ε-greedy policy. These policies enable agents to reach their destination with fewer steps, even without rewards. Additionally, direct learning agents following the softmax and inhibition-greedy policies outperform the agent using the ε-greedy policy. Nonetheless, during the rewarded phase, the latent learning agent utilizing the ε-greedy policy demonstrates a faster reduction in errors and exhibits quicker learning stabilization.

Notably, latent learning agents using the softmax policy in the Tolman setup exhibit greater instability in errors. While the precise reason for this instability is unclear, one could attribute to similar Q-values (see Fig. 35,b), which may introduce randomness action selection for this policy.

5.6 Summary

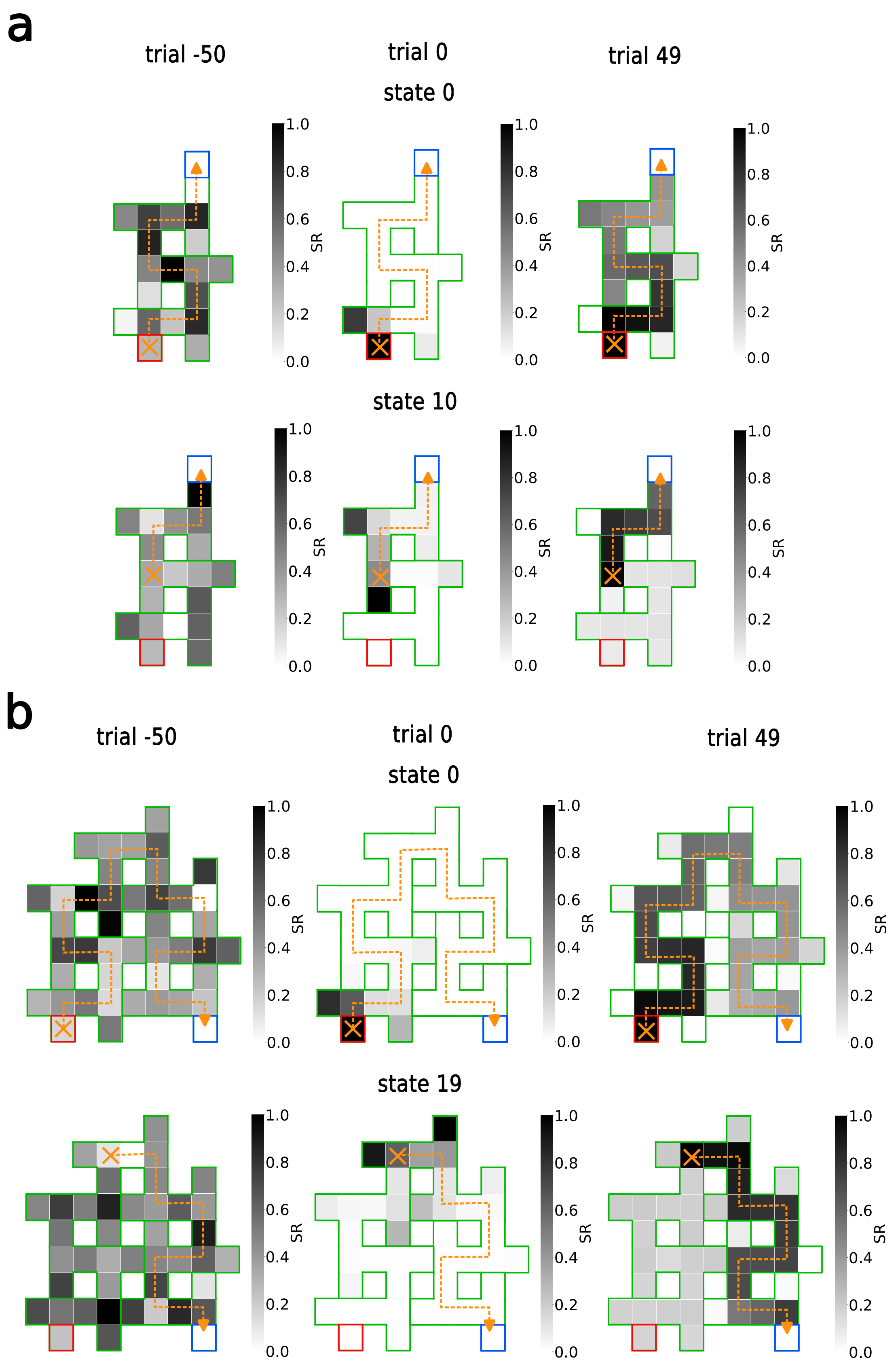
This chapter proposes a computational approach to mapping and learning in unfamiliar environments. Our proposed method was founded on combining CoBeL-RL and RatSLAM as an SRL algorithm. Firstly, we discussed how we integrated these frameworks and successfully validated this approach in a rewarded learning task. Further, we explored latent learning using this framework and conducted behavioral and structural analyses to understand how various exploration strategies and experiment designs can improve learning.

Figure 33 – Deep Successor Representation learned during Latent Learning.



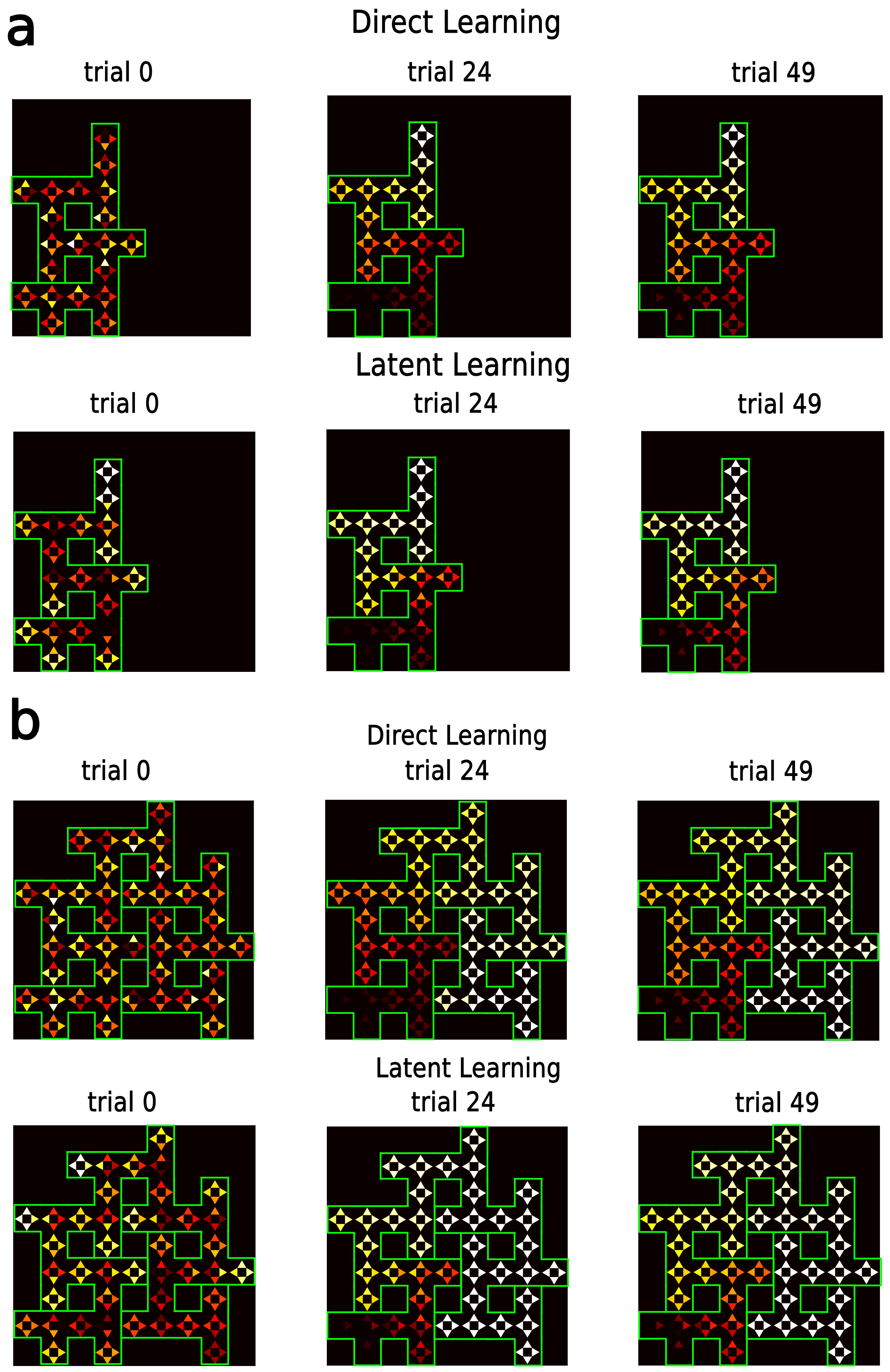
Source: Designed by the author

Figure 34 – Successor Representation row mapped to environment topology.



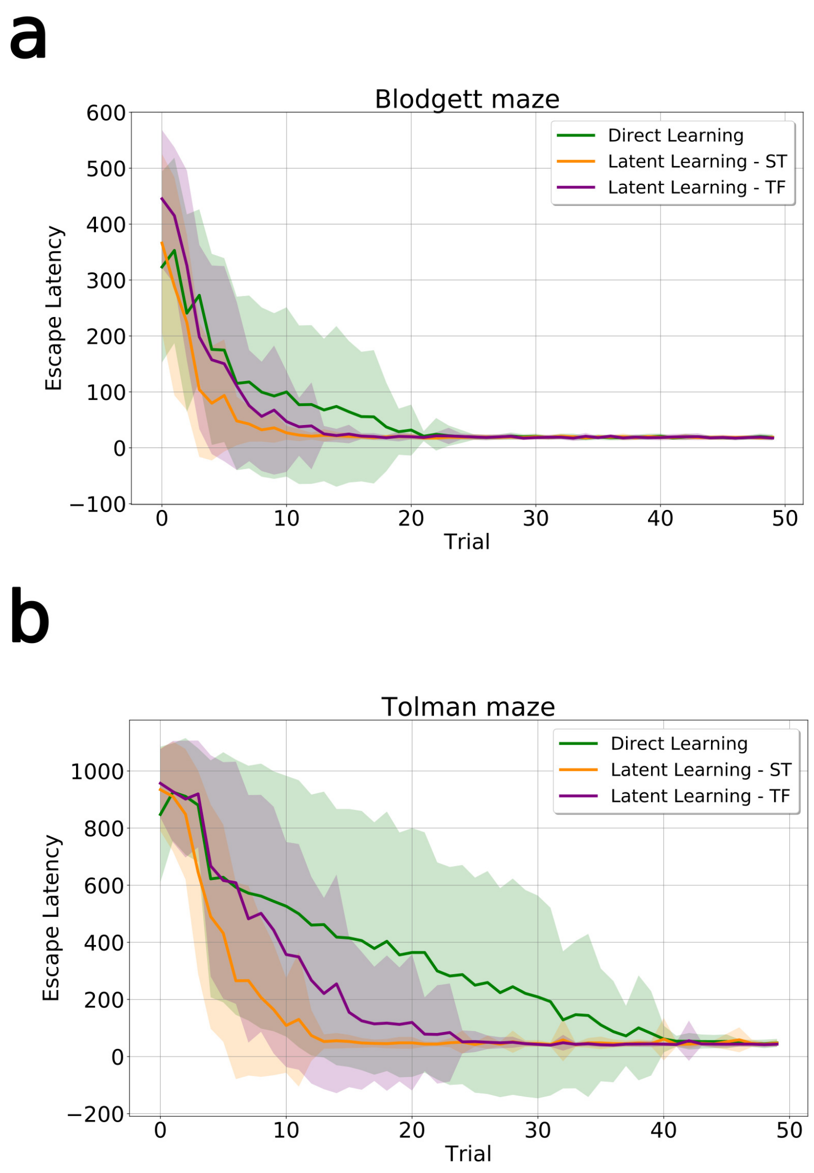
Source: Designed by the author

Figure 35 – Q-values in direct and latent learning agents.



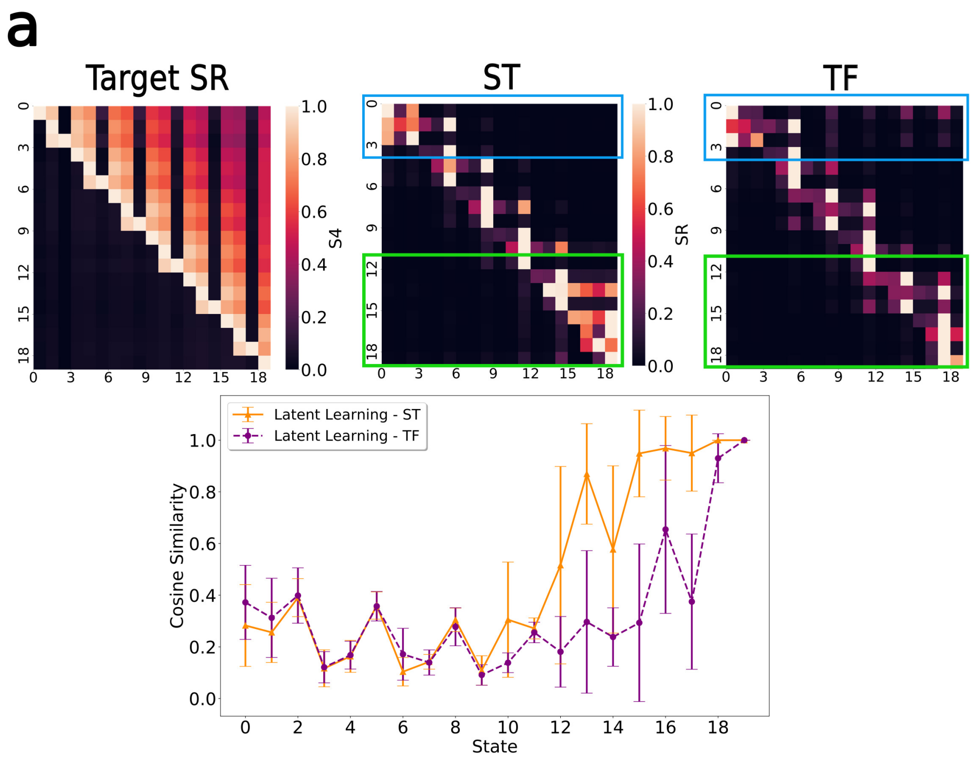
Source: Designed by the author

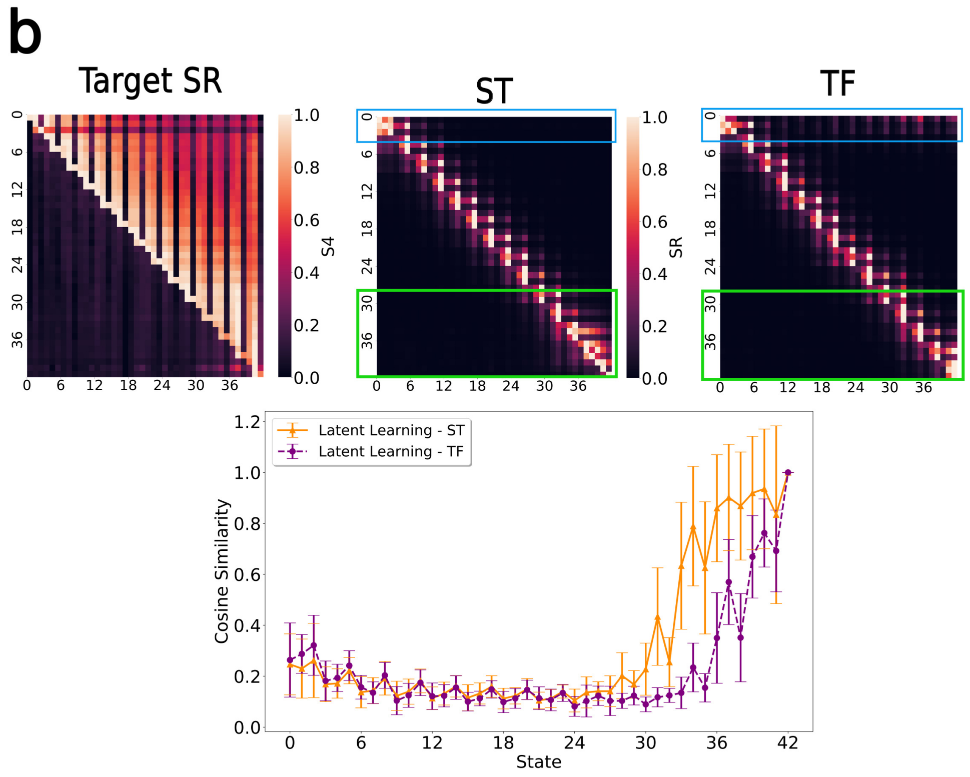
Figure 36 – Different Designs in Latent Learning.



Source: Designed by the author

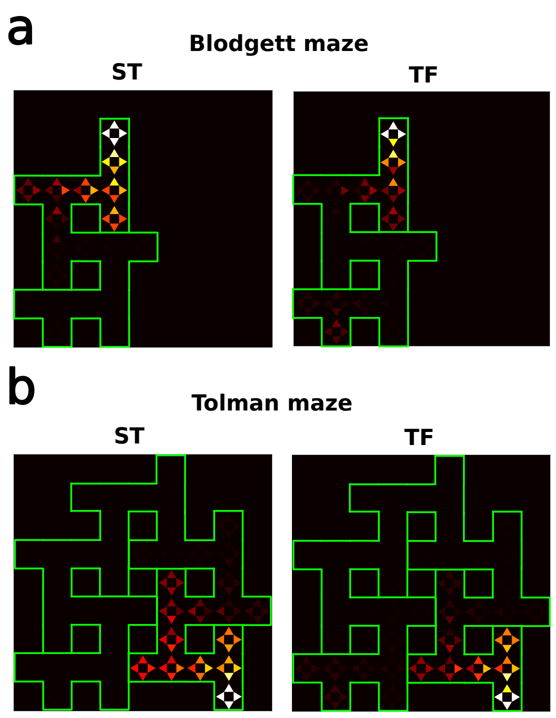
Figure 37 – Deep Successor Representation analysis over design experiments.





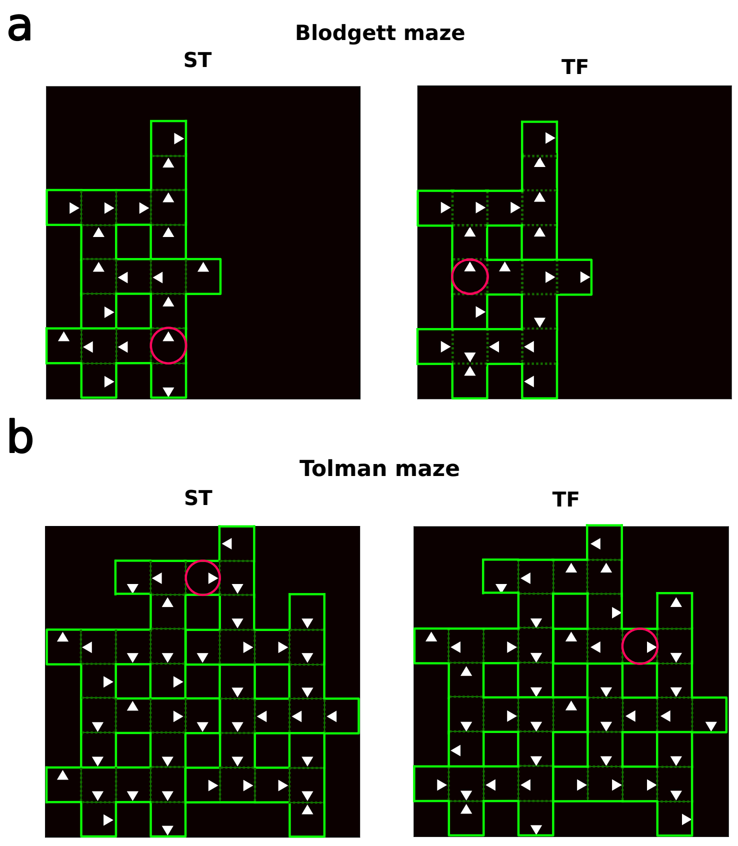
Source: Designed by the author

Figure 38 – Extracted Q-values from SR and ground-truth reward.



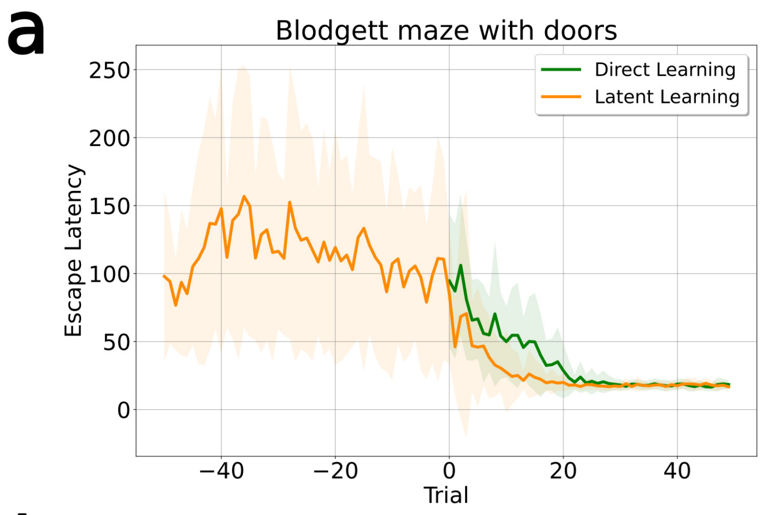
Source: Designed by the author

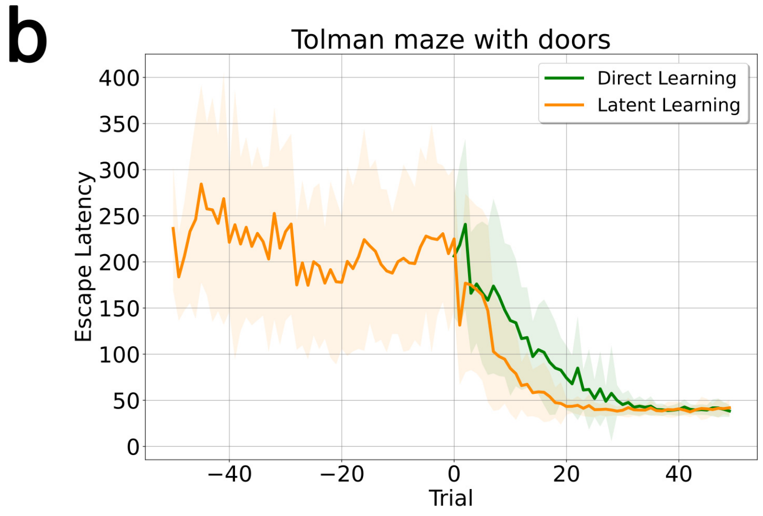
Figure 39 – Actions of the highest q-values for each state.



Source: Designed by the author

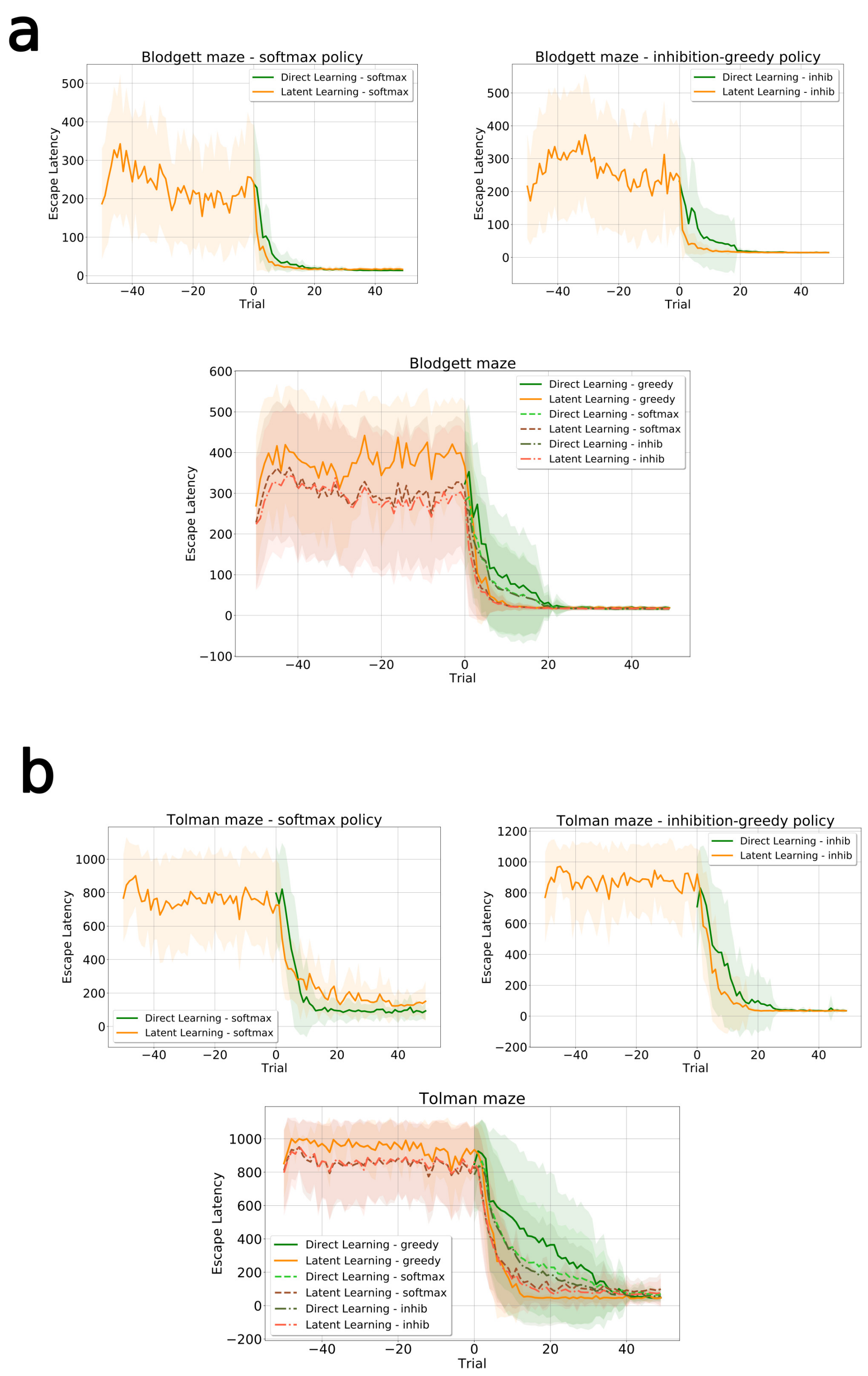
Figure 40 – Effect of doors in Latent Learning.





Source: Designed by the author

Figure 41 – Latent Learning in different RL policies.



Source: Designed by the author

6 Discussion and Conclusion

In this study, we propose a set of computational methods for building maps for navigation and spatial learning. Our approaches were built on two main frameworks, RatSLAM and CoBeL-RL. We performed improvements and research on two fronts: improving mapping capabilities for RatSLAM and combining RatSLAM and CoBeL-RL for mapping and learning in unfamiliar environments, which we have validated in latent learning experiments. We discuss the contributions in the following:

6.1 Enhancing Mapping with RatSLAM

Navigating unknown environments is a crucial yet challenging task involving distinct mechanisms within animals' brains. These mechanisms integrate external and internal information to determine the animal's position and devise navigation strategies. Furthermore, the spatial structure of the environment can influence how the internal cognitive map is represented in the brain. These fundamentals have facilitated the development of computational solutions for real-world problems, such as SLAM. Our base model for the computational cognitive map is RatSLAM, which shares essential mechanisms with animals, including visual calibration and path integration, in a simple and robust structure.

In this study, we extended the RatSLAM capabilities by proposing a method that enabled it to perform complex mapping tasks, such as multisession SLAM. This approach employed a merging mechanism that combines the map information from the previous session with the current one. What sets our method apart from existing ones is that, when encountering a new location, RatSLAM constructs a partial map and continues mapping instead of selecting the current map and continuing. If the partial map correlates with past maps, they are merged instead of discarding the current map and continuing the mapping process, as done in Tang, Yan e Tan (2018) and Milford e Wyeth (2010).

To allow multisession mapping in RatSLAM, a crucial aspect was the design of the merge mechanism that considers all RatSLAM frameworks. In particular, the Virtual Tour results revealed that ignoring the shift operation of LVC and PCN from the partial map after merging into the loaded map can result in incorrect associations between experiences during a loop closure. Consequently, this non-association generates an inaccurate representation of the global map.

Our multisession mapping results exhibited similarity with the maps generated through single session mapping in several experiments, e.g., *Virtual Tour* and *Lab Tour*. However, as the number of mapping sessions increased, there was a noticeable increase in path deviation, particularly in long-term and large-scale mapping scenarios. This tendency was evidenced by the ICP evaluations in the EMs of the *iRat* and *New College* experiments. The difference between the former may be due to the lack of connection between the inner and outer loop nodes. In the latter case, the path deviation occurred after an inconsistent path correction after the merge point.

Despite the presence of path distortions, it is notable that the final maps derived from both scenarios still maintained a significant resemblance to the single-session map, showing the potential use of the merge approach in multisession mapping that extends beyond two sessions and long-term mapping. Finally, our multisession approach corrected the Lab Tour map by approximating paths representing the same location after merging. The merge operations over distinct EMs allowed a more accurate global representation of the environment.

Aligned with multisession with a single agent mapping the environment, multisession SLAM algorithms can be adapted to operate with multiple agents. This configuration requires attention to the physical and temporal constraints of the multiple maps being constructed and merged during mapping. In this work, our multisession algorithm was not tested on multiple agents. However, additional research can be done on the core structure of the merge process to extend it for this type of SLAM configuration.

As future work, looking at the results we obtained from the *New College* dataset would be valuable. This exploration could help us improve the map merging process, especially in scenarios involving extensive or long-lasting environments. Another avenue for improvement could include upgrading the merging process to compare the ongoing mapping with a selection of previously stored maps in parallel. Such an enhancement would likely result in an algorithm that can adapt and perform well across various situations.

6.2 Enhancing Learning with Latent Learning

In addition to achieving complex mapping capabilities, animals can utilize their maps to improve spatial tasks, such as learning reward locations. Research has shown that these animals can enhance this learning through latent learning experiments. These findings are based on the cognitive map theory conceptualized by Tolman (1948) and supported by subsequent discoveries of brain cells. Leveraging this understanding, the advantage of first acquiring a map of an environment becomes particularly evident in applications like search-and-rescue missions. By learning the environment's layout without explicit motivation and before engaging in specific tasks, agents or organisms can navigate more efficiently and effectively, strategically locating targets or objectives without extensive trial and error. This preliminary map-based navigation minimizes uncertainty and optimizes decision-making, streamlining the entire process and potentially saving valuable time in critical situations.

Recently, a new hypothesis regarding the nature of the cognitive map has emerged, suggesting its predictive nature. This hypothesis aligns with learning a representation of the environment. The Successor Representation (SR) algorithm within the Reinforcement Learning (RL) framework can learn such a predictive model. However, existing frameworks for latent learning often rely on a pre-existing model of the environment, which fails to account for a biologically plausible model when the agent needs to learn from an unknown environment.

As a relevant contribution to this thesis, we propose an alternative approach by integrating RatSLAM with the CoBeL-RL framework to address the learning of states (mapping) for RL agents in unknown environments. By adding a mapping mechanism into the RL framework, researchers might have more flexibility to investigate paradigms such as exploration-exploitation. They could build realistic-inspired scenarios for learning in computational setups. Here, the RatSLAM worked as the State Representation Learning (SRL), which allowed an SR agent to simultaneously build a map of the environment and learn during this process. Furthermore, the framework's integration introduced a new variable for true exploration (τ). The (τ) value defines how the agent would explore new areas of environments, in which agents with $\tau = 1$ prioritize discovering new places and, therefore, learn a new state for the environment. Adding τ can be associated with how "curious" (artificial curiosity) the agent might be about the unexplored locations.

During the simulation of a rewarded task, it was observed that agents with $\tau = [0.5, 0.1]$ did not manage to learn the task optimally in the Tolman maze configuration. Two notable observations were made concerning these results. First, with $\tau < 1.0$, the stochastic nature of the simulations introduced the possibility that agents might not be able to find the location of the reward. On the other hand, agents with a higher probability of finding the target's location, such as those with $\tau = 1.0$, learn the task more quickly, mainly in the initial attempts and in an optimized way. In other words, the more curious the agent was about new areas of the environment, the faster the acquisition of states associated with the environment, and, as demonstrated by the results of our experiments, these agents learned the rewarded task optimally.

Moreover, we demonstrated the compatibility of efficient learning with findings in neuroscience upon different designs and exploration strategies in latent learning experiments, such as the works of (KARN; JR., 1946) and (DAUB, 1933). In investigating behavior during latent learning experiments, our findings suggested that the agent's design experiments and exploration strategy impact the learning speed among different groups. For instance, experiment designs in the unrewarded phase that closely resemble the rewarded tasks yield the most significant improvement in learning when the motivation is presented in the environment.

This study also investigated different exploration strategies in latent learning. Our results have shown that the differences in learning speed might be attributed to how agents explore the environment. The inhibition-greedy policy proposed in our study offers distinct advantages over the softmax or ε-greedy policies. Specifically, in the context of Tolman's setup, the escape latency results revealed that our proposed policy outperformed the ε-greedy policy employed by the direct learning agents and demonstrated improved learning stability for the latent learning agents compared to the softmax policy. These findings highlight the potential of our proposed policy as a solution for addressing the exploration-exploitation trade-off in reinforcement learning.

Future research in this area may investigate the potential applications of the integrated RatSLAM and CoBeL-RL framework in real-world scenarios, including robotics and autonomous navigation. Additionally, exploring the impact of environmental factors on latent learning and spatial memory could lead to a better understanding of how animals and agents navigate their surroundings. Further investigation into the predictive nature of cognitive maps could also provide insights into how these maps are formed and utilized in decision-making processes.

Ultimately, these findings contribute to advancing computational techniques in building maps for spatial navigation and learning, holding potential for real-world applications in the robotic and neuroscience fields.

6.3 Scientific Productions

Table 3 presents the published articles directly related to the proposed method and as the first author. Additionally, Table 4 lists the scientific papers published and submitted as the co-author related to the proposed method in spatial navigation applications since the beginning of the doctoral program.

Table 3 – Scientific productions related to the proposed methods as the first author.

Article	**Type**	**Qualis**	**Status**
A Multisession SLAM Approach for RatSLAM In: Journal of Intelligent & Robotic Systems Year: 2023.	Journal	A2	Published
Automatic Tuning of RatSLAM's Parameters by Irace and Iterative Closest Point In: IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society Year: 2020.	Conference	A2	Published

Table 4 – Scientific productions related to the proposed methods and the spatial navigation field as the co-author.

Articles	**Type**	**Qualis**	**Status**
CoBeL-RL: A Neuroscience-oriented Simulation Framework for Complex Behavior and Learning In: Frontiers in Neuroinformatics Year: 2023.	Journal	A2	Published
XRatSLAM: An Extensible RatSLAM Computational Framework In: Sensors Year: 2022.	Journal	A2	Published
Loss function regularization on the Iterated Racing Procedure for Automatic Tuning of RatSLAM Parameters. In: Computational Neuroscience, Series of Communications in Computer and Information Science, Third Latin American Workshop, LAWCN Year: 2021.	Book Chapter	B2	Published

Bibliography

POULTER, S.; HARTLEY, T.; LEVER, C. The neurobiology of mammalian navigation. **Current Biology**, v. 28, n. 17, p. R1023–R1042, 2018. ISSN 0960-9822. Available at: <https://www.sciencedirect.com/science/article/pii/S0960982218306869>. Cited 2 times on pages 15 and 43.

HILLS, T. T.; TODD, P. M.; LAZER, D.; REDISH, A. D.; COUZIN, I. D. Exploration versus exploitation in space, mind, and society. **Trends in Cognitive Sciences**, v. 19, n. 1, p. 46–54, 2015. ISSN 1364-6613. Available at: <https://www.sciencedirect.com/science/article/pii/S1364661314002332>. Cited on page 15.

O'KEEFE, J.; NADEL, L. **The hippocampus as a cognitive map**. Oxford, United Kingdom: Clarendon Press, 1978. – p. Cited on page 15.

TAUBE, J.; MULLER, R.; RANCK, J. J. Head-direction cells recorded from the post-subiculum in freely moving rats. i. description and quantitative analysis. **Journal of Neuroscience**, Society for Neuroscience, v. 10, n. 2, p. 420–435, 1990. ISSN 0270-6474. Available at: <https://www.jneurosci.org/content/10/2/420>. Cited 2 times on pages 15 and 16.

HAFTING, T.; FYHN, M.; MOLDEN, S.; MOSER, M.-B.; MOSER, E. I. Microstructure of a spatial map in the entorhinal cortex. **Nature**, v. 436, n. 7052, p. 801–806, Aug 2005. ISSN 1476-4687. Available at: <https://doi.org/10.1038/nature03721>. Cited 2 times on pages 15 and 16.

BEHRENS, T. E.; MULLER, T. H.; WHITTINGTON, J. C.; MARK, S.; BARAM, A. B.; STACHENFELD, K. L.; KURTH-NELSON, Z. What is a cognitive map? organizing knowledge for flexible behavior. **Neuron**, v. 100, n. 2, p. 490–509, 2018. ISSN 0896-6273. Available at: <https://www.sciencedirect.com/science/article/pii/S0896627318308560>. Cited 2 times on pages 15 and 16.

JIN, W.; QIN, H.; ZHANG, K.; CHEN, X. Spatial navigation. In: _____. **Neural Circuits of Innate Behaviors**. Singapore: Springer Singapore, 2020. p. 63–90. ISBN 978-981-15-7086-5. Available at: <https://doi.org/10.1007/978-981-15-7086-5_7>. Cited on page 15.

TOLMAN, E. C. Cognitive maps in rats and men. **Psychological Review**, American Psychological Association, US, v. 55, p. 189–208, 1948. Available at: <https://doi.org/10.1037/h0061626>. Cited 8 times on pages 15, 42, 43, 45, 75, 77, 79, and 95.

TOLMAN, E. C.; HONZIK, C. H. Introduction and removal of reward, and maze performance in rats. **University of California Publications in Psychology**, v. 4, p. 257–275, 1930. Cited on page 15.

CARPENTER, F.; MANSON, D.; JEFFERY, K.; BURGESS, N.; BARRY, C. Grid cells form a global representation of connected environments. **Current Biology**, Elsevier, v. 25, n. 9, p. 1176–1182, May 2015. ISSN 0960-9822. Available at: <https://doi.org/10.1016/j.cub.2015.02.037>. Cited 2 times on pages 16 and 19.

WERNLE, T.; WAAGA, T.; MØRREAUNET, M.; TREVES, A.; MOSER, M.-B.; MOSER, E. I. Integration of grid maps in merged environments. **Nature Neuroscience**, v. 21, n. 1, p. 92–101, Jan 2018. ISSN 1546-1726. Available at: <https://doi.org/10.1038/s41593-017-0036-6>. Cited 3 times on pages 16, 19, and 40.

STACHENFELD, K. L.; BOTVINICK, M. M.; GERSHMAN, S. J. The hippocampus as a predictive map. **Nature Neuroscience**, v. 20, n. 11, p. 1643–1653, Nov 2017. ISSN 1546-1726. Available at: <https://doi.org/10.1038/nn.4650>. Cited 4 times on pages 16, 17, 33, and 44.

de Cothi, W.; NYBERG, N.; GRIESBAUER, E.-M.; GHANAMé, C.; ZISCH, F.; LEFORT, J. M.; FLETCHER, L.; NEWTON, C.; RENAUDINEAU, S.; BENDOR, D.; GRIEVES, R.; DUVELLE Éléonore; BARRY, C.; SPIERS, H. J. Predictive maps in rats and humans for spatial navigation. **Current Biology**, v. 32, n. 17, p. 3676–3689.e5, 2022. ISSN 0960-9822. Available at: <https://www.sciencedirect.com/science/article/pii/S0960982222010958>. Cited 4 times on pages 16, 17, 18, and 44.

BRUNEC, I. K.; MOMENNEJAD, I. Predictive representations in hippocampal and prefrontal hierarchies. **Journal of Neuroscience**, Society for Neuroscience, v. 42, n. 2, p. 299–312, 2022. ISSN 0270-6474. Available at: <https://www.jneurosci.org/content/42/2/299>. Cited 2 times on pages 16 and 17.

KARN, H. W.; JR., J. M. P. The effects of certain pre-training procedures upon maze performance and their significance for the concept of latent learning. **Journal of Experimental Psychology**, American Psychological Association, US, v. 36, p. 461–469, 1946. ISSN 0022-1015(Print). Available at: <https://doi.org/10.1037/h0061422>. Cited 7 times on pages 17, 19, 46, 72, 81, 82, and 96.

ZENO, P. J.; PATEL, S.; SOBH, T. M. Review of neurobiologically based mobile robot navigation system research performed since 2000. **J. Robot.**, Hindawi Limited, London, GBR, v. 2016, p. 2, sep 2016. ISSN 1687-9600. Available at: <https://doi.org/10.1155/2016/8637251>. Cited on page 17.

TANG, H.; YAN, R.; TAN, K. C. Cognitive navigation by neuro-inspired localization, mapping, and episodic memory. **IEEE Transactions on Cognitive and Developmental Systems**, v. 10, n. 3, p. 751–761, 2018. Cited 5 times on pages 17, 39, 40, 41, and 94.

YU, F.; SHANG, J.; HU, Y.; MILFORD, M. Neuroslam: a brain-inspired slam system for 3d environments. **Biological Cybernetics**, v. 113, n. 5, p. 515–545, Dec 2019. ISSN 1432-0770. Available at: <https://doi.org/10.1007/s00422-019-00806-9>. Cited 2 times on pages 17 and 39.

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localization and mapping: part i. **IEEE Robotics & Automation Magazine**, v. 13, n. 2, p. 99–110, 2006. Cited 2 times on pages 17 and 20.

MILFORD, M.; WYETH, G.; PRASSER, D. Ratslam: a hippocampal model for simultaneous localization and mapping. In: **IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004**. [S.l.: s.n.], 2004. v. 1, p. 403–408 Vol.1. Cited 3 times on pages 17, 21, and 39.

MILFORD, M. J.; WILES, J.; WYETH, G. F. Solving navigational uncertainty using grid cells on robots. **PLOS Computational Biology**, Public Library of Science, v. 6, n. 11, p. 1–14, 11 2010. Available at: <https://doi.org/10.1371/journal.pcbi.1000995>. Cited 4 times on pages 17, 23, 24, and 39.

BALL, D.; HEATH, S.; WILES, J.; WYETH, G.; CORKE, P.; MILFORD, M. Openratslam: an open source brain-based slam system. **Autonomous Robots**, v. 34, n. 3, p. 149–176, Apr 2013. ISSN 1573-7527. Available at: <https://doi.org/10.1007/s10514-012-9317-9>. Cited 13 times on pages 17, 21, 23, 24, 25, 39, 40, 53, 54, 56, 57, 59, and 60.

TESSEREAU, C.; O'DEA, R.; COOMBES, S.; BAST, T. Reinforcement learning approaches to hippocampus-dependent flexible spatial navigation. **Brain and Neuroscience Advances**, v. 5, p. 2398212820975634, 2021. PMID: 33954259. Available at: <https://doi.org/10.1177/2398212820975634>. Cited on page 18.

HE, Q.; LIU, J. L.; ESCHAPASSE, L.; BEVERIDGE, E. H.; BROWN, T. I. A comparison of reinforcement learning models of human spatial navigation. **Scientific Reports**, v. 12, n. 1, p. 13923, Aug 2022. ISSN 2045-2322. Available at: <https://doi.org/10.1038/s41598-022-18245-1>. Cited on page 18.

DIEKMANN, N.; VIJAYABASKARAN, S.; ZENG, X.; KAPPEL, D.; MENEZES, M. C.; CHENG, S. Cobel-rl: A neuroscience-oriented simulation framework for complex behavior and learning. **Frontiers in Neuroinformatics**, v. 17, 2023. Cited 7 times on pages 18, 19, 35, 36, 44, 54, and 73.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. Cambridge, MA, USA: A Bradford Book, 2018. ISBN 0262039249. Cited 9 times on pages 18, 26, 27, 28, 29, 30, 31, 37, and 83.

LESORT, T.; DíAZ-RODRíGUEZ, N.; GOUDOU, J.-F.; FILLIAT, D. State representation learning for control: An overview. **Neural Networks**, v. 108, p. 379–392, 2018. ISSN 0893-6080. Available at: <https://www.sciencedirect.com/science/article/pii/S0893608018302053>. Cited 2 times on pages 18 and 44.

MERCKLING, A.; PERRIN-GILBERT, N.; CONINX, A.; DONCIEUX, S. Exploratory state representation learning. **Frontiers in Robotics and AI**, v. 9, 2022. ISSN 2296-9144. Available at: <https://www.frontiersin.org/articles/10.3389/frobt.2022.762051>. Cited on page 18.

DAYAN, P. Improving Generalization for Temporal Difference Learning: The Successor Representation. **Neural Computation**, v. 5, n. 4, p. 613–624, 07 1993. ISSN 0899-7667. Available at: <https://doi.org/10.1162/neco.1993.5.4.613>. Cited 3 times on pages 18, 32, and 44.

RUSSEK, E. M.; MOMENNEJAD, I.; BOTVINICK, M. M.; GERSHMAN, S. J.; DAW, N. D. Predictive representations can link model-based reinforcement learning to model-free mechanisms. **PLOS Computational Biology**, Public Library of Science, v. 13, n. 9, p. 1–35, 09 2017. Available at: <https://doi.org/10.1371/journal.pcbi.1005768>. Cited 2 times on pages 18 and 44.

DUCAROUGE, A.; SIGAUD, O. The Successor Representation as a model of behavioural flexibility. In: **Journées Francophones sur la Planification, la Décision et**

l'Apprentissage pour la conduite de systèmes (JFPDA 2017). Caen, France: [s.n.], 2017. (Actes des Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes (JFPDA 2017)). Available at: <https://hal.science/hal-01576352>. Cited 2 times on pages 18 and 44.

BLODGETT, H. C. The effect of the introduction of reward upon the maze performance of rats. **University of California Publications in Psychology**, v. 4, p. 113–134, 1929. Cited 3 times on pages 18, 42, and 44.

PATHAK, D.; AGRAWAL, P.; EFROS, A. A.; DARRELL, T. Curiosity-driven exploration by self-supervised prediction. In: PMLR. **International conference on machine learning**. [S.l.], 2017. p. 2778–2787. Cited 2 times on pages 18 and 74.

MENEZES, M.; MUÑOZ, M.; FREITAS, E. Pignaton de; CHENG, S.; NETO, A. de A.; RIBEIRO, P.; OLIVEIRA, A. A multisession slam approach for ratslam. **Journal of Intelligent & Robotic Systems**, v. 108, n. 4, p. 61, Jul 2023. ISSN 1573-0409. Available at: <https://doi.org/10.1007/s10846-023-01816-3>. Cited 16 times on pages 19, 22, 23, 49, 50, 56, 57, 58, 59, 64, 65, 66, 67, 68, 69, and 70.

MENEZES, M. C.; MUñOZ, M. E. S.; FREITAS, E. P.; CHENG, S.; WALTHER, T.; NETO, A. A.; RIBEIRO, P. R. A.; OLIVEIRA, A. C. M. Automatic tuning of ratslam's parameters by irace and iterative closest point. In: **IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society**. [S.l.: s.n.], 2020. p. 562–568. Cited 2 times on pages 19 and 60.

GOMES, P. G. B.; OLIVEIRA, C. J. R. L. de; MENEZES, M. C.; RIBEIRO, P. R. d. A.; OLIVEIRA, A. C. M. de. Loss function regularization on the iterated racing procedure for automatic tuning of ratslam parameters. In: RIBEIRO, P. R. d. A.; COTA, V. R.; BARONE, D. A. C.; OLIVEIRA, A. C. M. de (Ed.). **Computational Neuroscience**. Cham: Springer International Publishing, 2022. p. 48–63. ISBN 978-3-031-08443-0. Cited 3 times on pages 19, 60, and 62.

THRUN, S. Simultaneous localization and mapping. In: ____. **Robotics and Cognitive Approaches to Spatial Mapping**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 13–41. ISBN 978-3-540-75388-9. Available at: <https://doi.org/10.1007/978-3-540-75388-9_3>. Cited 2 times on pages 20 and 21.

ZAFFAR, M.; EHSAN, S.; STOLKIN, R.; MAIER, K. M. Sensors, slam and long-term autonomy: A review. In: **2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)**. [S.l.: s.n.], 2018. p. 285–290. Cited on page 20.

MONTEMERLO, M.; THRUN, S.; ROLLER, D.; WEGBREIT, B. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: **Proceedings of the 18th International Joint Conference on Artificial Intelligence**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003. (IJCAI'03), p. 1151–1156. Cited on page 20.

GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improved techniques for grid mapping with rao-blackwellized particle filters. **IEEE Transactions on Robotics**, v. 23, n. 1, p. 34–46, 2007. Cited on page 20.

SUBRAMANIAN, A.; CHITLANGIA, S.; BATHS, V. Reinforcement learning and its connections with neuroscience and psychology. **Neural Networks**, v. 145, p. 271–287, 2022. ISSN 0893-6080. Available at: <https://www.sciencedirect.com/science/article/pii/S0893608021003944>. Cited on page 26.

SCHULTZ, W. Multiple reward signals in the brain. **Nature Reviews Neuroscience**, v. 1, n. 3, p. 199–207, Dec 2000. ISSN 1471-0048. Available at: <https://doi.org/10.1038/35044563>. Cited on page 26.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLEMARE, M. G.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A. K.; OSTROVSKI, G.; PETERSEN, S.; BEATTIE, C.; SADIK, A.; ANTONOGLOU, I.; KING, H.; KUMARAN, D.; WIERSTRA, D.; LEGG, S.; HASSABIS, D. Human-level control through deep reinforcement learning. **Nature**, v. 518, n. 7540, p. 529–533, Feb 2015. ISSN 1476-4687. Available at: <https://doi.org/10.1038/nature14236>. Cited 2 times on pages 31 and 36.

KULKARNI, T. D.; SAEEDI, A.; GAUTAM, S.; GERSHMAN, S. J. **Deep Successor Reinforcement Learning**. 2016. Cited 2 times on pages 34 and 37.

Blender Online Community. **Blender is the free and open source 3D creation suite.** Blender Institute, Amsterdam, 2018. Available at: <http://www.blender.org>. Cited on page 35.

LINIETSKY, J.; MANZUR, A. **Godot Engine**. [S.l.], 2007. Available at: <https://godotengine.org>. Cited on page 35.

Unity Technologies. **Unity**. [S.l.], 2005. Available at: <https://unity.com/>. Cited on page 35.

LIN, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. **Machine Learning**, v. 8, n. 3, p. 293–321, May 1992. ISSN 1573-0565. Available at: <https://doi.org/10.1007/BF00992699>. Cited on page 36.

SCHAUL, T.; QUAN, J.; ANTONOGLOU, I.; SILVER, D. **Prioritized Experience Replay**. 2016. Cited on page 37.

WANG, J.; YAN, R.; TANG, H. Multi-scale extension in an entorhinal-hippocampal model for cognitive map building. **Frontiers in Neurorobotics**, v. 14, 2021. ISSN 1662-5218. Available at: <https://www.frontiersin.org/articles/10.3389/fnbot.2020.592057>. Cited on page 39.

ZENG, T.; SI, B. Cognitive mapping based on conjunctive representations of space and movement. **Frontiers in neurorobotics**, Frontiers Media SA, v. 11, p. 61, 2017. Cited on page 39.

ZENG, T.; TANG, F.; JI, D.; SI, B. Neurobayesslam: Neurobiologically inspired bayesian integration of multisensory information for robot navigation. **Neural Networks**, v. 126, p. 21–35, 2020. ISSN 0893-6080. Available at: <https://www.sciencedirect.com/science/article/pii/S0893608020300770>. Cited on page 39.

PEER, M.; BRUNEC, I. K.; NEWCOMBE, N. S.; EPSTEIN, R. A. Structuring knowledge with cognitive maps and cognitive graphs. **Trends in Cognitive Sciences**, v. 25, n. 1, p. 37–54, 2021. ISSN 1364-6613. Available at: <https://www.sciencedirect.com/science/article/pii/S1364661320302503>. Cited on page 39.

PRASSER, D.; MILFORD, M.; WYETH, G. Outdoor simultaneous localisation and mapping using ratslam. In: CORKE, P.; SUKKARIAH, S. (Ed.). **Field and Service Robotics**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 143–154. ISBN 978-3-540-33453-8. Cited on page 40.

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, p. 1038–1053, 2008. Cited 2 times on pages 40 and 54.

MILFORD, M.; WYETH, G. Persistent navigation and mapping using a biologically inspired slam system. **The International Journal of Robotics Research**, v. 29, n. 9, p. 1131–1153, 2010. Available at: <https://doi.org/10.1177/0278364909340592>. Cited 2 times on pages 40 and 94.

MENEZES, M. C.; FREITAS, E. P. de; CHENG, S.; OLIVEIRA, A. C. M. de; RIBEIRO, P. R. de A. A neuro-inspired approach to solve a simultaneous location and mapping task using shared information in multiple robots systems. In: **2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)**. [S.l.: s.n.], 2018. p. 1753–1758. Cited on page 40.

MCDONALD, J.; KAESS, M.; CADENA, C.; NEIRA, J.; LEONARD, J. Real-time 6-dof multi-session visual slam over large-scale environments. **Robotics and Autonomous Systems**, v. 61, n. 10, p. 1144–1158, 2013. ISSN 0921-8890. Selected Papers from the 5th European Conference on Mobile Robots (ECMR 2011). Available at: <https://www.sciencedirect.com/science/article/pii/S0921889012001406>. Cited 2 times on pages 40 and 41.

LABBÉ, M.; MICHAUD, F. Long-term online multi-session graph-based splam with memory management. **Autonomous Robots**, v. 42, n. 6, p. 1133–1150, Aug 2018. ISSN 1573-7527. Available at: <https://doi.org/10.1007/s10514-017-9682-5>. Cited 2 times on pages 40 and 41.

LABBé, M.; MICHAUD, F. Online global loop closure detection for large-scale multi-session graph-based slam. In: **2014 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2014. p. 2661–2666. Cited on page 41.

WANG, Y.; HUANG, S.; XIONG, R.; WU, J. A framework for multi-session rgbd slam in low dynamic workspace environment. **CAAI Transactions on Intelligence Technology**, v. 1, n. 1, p. 90–103, 2016. ISSN 2468-2322. Available at: <https://www.sciencedirect.com/science/article/pii/S246823221600010X>. Cited on page 41.

DAOUD, H. A.; SABRI, A. Q. M.; LOO, C. K.; MANSOOR, A. M. Slamm: Visual monocular slam with continuous mapping using multiple maps. **PLOS ONE**, Public Library of Science, v. 13, n. 4, p. 1–22, 04 2018. Available at: <https://doi.org/10.1371/journal.pone.0195878>. Cited on page 41.

SCHNEIDER, T.; DYMCZYK, M.; FEHR, M.; EGGER, K.; LYNEN, S.; GILITSCHENSKI, I.; SIEGWART, R. Maplab: An open framework for research in visual-inertial mapping and localization. **IEEE Robotics and Automation Letters**, v. 3, n. 3, p. 1418–1425, 2018. Cited on page 41.

BURGUERA, A. B.; BONIN-FONT, F. A trajectory-based approach to multi-session underwater visual slam using global image signatures. **Journal of Marine Science and Engineering**, v. 7, n. 8, 2019. ISSN 2077-1312. Available at: <https://www.mdpi.com/2077-1312/7/8/278>. Cited on page 41.

CAMPOS, C.; ELVIRA, R.; RODRíGUEZ, J. J. G.; MONTIEL, J. M. M.; TARDóS, J. D. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. **IEEE Transactions on Robotics**, v. 37, n. 6, p. 1874–1890, 2021. Cited on page 41.

LABBé, M.; MICHAUD, F. Multi-session visual slam for illumination-invariant re-localization in indoor environments. **Frontiers in Robotics and AI**, v. 9, 2022. ISSN 2296-9144. Available at: <https://www.frontiersin.org/articles/10.3389/frobt.2022.801886>. Cited on page 41.

TOLMAN, E. C.; HONZIK, C. H. Introduction and removal of reward, and maze performance in rats. **University of California Publications in Psychology**, v. 4, p. 257–275, 1930. Cited on page 42.

SUTHERLAND, R. J.; LINGGARD, R. Being there: a novel demonstration of latent spatial learning in the rat. **Behavioral and Neural Biology**, v. 36, n. 2, p. 103–107, 1982. ISSN 0163-1047. Available at: <https://www.sciencedirect.com/science/article/pii/S0163104782901017>. Cited 2 times on pages 42 and 46.

SUTHERLAND, R. J.; CHEW, G. L.; BAKER, J. C.; LINGGARD, R. C. Some limitations on the use of distal cues in place navigation by rats. **Psychobiology**, v. 15, n. 1, p. 48–57, Mar 1987. ISSN 0889-6313. Available at: <https://doi.org/10.3758/BF03327263>. Cited on page 42.

KEITH, J. R.; MCVETY, K. M. Latent place learning in a novel environment and the influences of prior training in rats. **Psychobiology**, v. 16, n. 2, p. 146–151, Jun 1988. ISSN 0889-6313. Available at: <https://doi.org/10.3758/BF03333116>. Cited on page 42.

KIMBLE, D. P.; GREENE, E. G. Absence of latent learning in rats with hippocampal lesions. **Psychonomic Science**, v. 11, n. 3, p. 99–100, Mar 1968. ISSN 0033-3131. Available at: <https://doi.org/10.3758/BF03328154>. Cited on page 42.

KIMBLE, D. P.; BREMILLER, R. Latent learning in hippocampal-lesioned rats. **Physiology & Behavior**, v. 26, n. 6, p. 1055–1059, 1981. ISSN 0031-9384. Available at: <https://www.sciencedirect.com/science/article/pii/0031938481902092>. Cited on page 42.

MEANS, L. W. Latent learning in rats with hippocampal lesions. **Psychonomic Science**, v. 16, n. 1, p. 51–52, Jan 1969. ISSN 0033-3131. Available at: <https://doi.org/10.3758/BF03331911>. Cited on page 42.

OWEN, M. J.; BUTLER, S. The effects of fornix lesions on latent learning in the rat. **Physiology & Behavior**, v. 24, n. 5, p. 817–822, 1980. ISSN 0031-9384. Available at: <https://www.sciencedirect.com/science/article/pii/003193848090133X>. Cited on page 42.

KIMBLE, D. P.; JORDAN, W. P.; BREMILLER, R. Further evidence for latent learning in hippocampal-lesioned rats. **Physiology & Behavior**, v. 29, n. 3, p. 401–407, 1982. ISSN 0031-9384. Available at: <https://www.sciencedirect.com/science/article/pii/003193848290258X>. Cited on page 42.

GUO, W.; ZHANG, J. J.; NEWMAN, J. P.; WILSON, M. A. Latent learning drives sleep-dependent plasticity in distinct ca1 subpopulations. **bioRxiv**, Cold Spring Harbor Laboratory, 2020. Available at: <https://www.biorxiv.org/content/early/2020/02/27/2020.02.27.967794>. Cited on page 42.

BARROS, A. C. B. de; BARUCHIN, L. J.; PANAYI, M. C.; NYBERG, N.; SAMBORSKA, V.; MEALING, M. T.; AKAM, T.; KWAG, J.; BANNERMAN, D. M.; KOHL, M. M. Retrosplenial cortex is necessary for spatial and non-spatial latent learning in mice. **bioRxiv**, Cold Spring Harbor Laboratory, 2021. Available at: <https://www.biorxiv.org/content/early/2021/07/23/2021.07.21.453258>. Cited on page 42.

THISTLETHWAITE, D. A critical review of latent learning and related experiments. **Psychological Bulletin**, American Psychological Association, US, v. 48, p. 97–129, 1951. Available at: <https://doi.org/10.1037/h0055171>. Cited 3 times on pages 42, 45, and 81.

PAVLIDES, C.; WINSON, J. Influences of hippocampal place cell firing in the awake state on the activity of these cells during subsequent sleep episodes. **Journal of Neuroscience**, Society for Neuroscience, v. 9, n. 8, p. 2907–2918, 1989. ISSN 0270-6474. Available at: <https://www.jneurosci.org/content/9/8/2907>. Cited on page 43.

KARLSSON, M. P.; FRANK, L. M. Awake replay of remote experiences in the hippocampus. **Nature Neuroscience**, v. 12, n. 7, p. 913–918, Jul 2009. ISSN 1546-1726. Available at: <https://doi.org/10.1038/nn.2344>. Cited on page 43.

ÓLAFSDÓTTIR, H. F.; CARPENTER, F.; BARRY, C. Coordinated grid and place cell replay during rest. **Nature Neuroscience**, v. 19, n. 6, p. 792–794, Jun 2016. ISSN 1546-1726. Available at: <https://doi.org/10.1038/nn.4291>. Cited on page 43.

MOMENNEJAD, I. Learning structures: Predictive representations, replay, and generalization. **Current Opinion in Behavioral Sciences**, v. 32, p. 155–166, 2020. ISSN 2352-1546. Understanding memory: Which level of analysis? Available at: <https://www.sciencedirect.com/science/article/pii/S2352154620300371>. Cited on page 43.

SCLEIDOROVICH, P.; LLOFRIU, M.; FELLOUS, J.-M.; WEITZENFELD, A. A computational model for latent learning based on hippocampal replay. In: **2020 International Joint Conference on Neural Networks (IJCNN)**. [S.l.: s.n.], 2020. p. 1–8. Cited 2 times on pages 43 and 82.

DIEKMANN, N.; CHENG, S. A model of hippocampal replay driven by experience and environmental structure facilitates spatial learning. **eLife**, eLife Sciences Publications, Ltd, v. 12, p. e82301, mar 2023. ISSN 2050-084X. Available at: <https://doi.org/10.7554/eLife.82301>. Cited on page 43.

GERSHMAN, S. J. The successor representation: Its computational logic and neural substrates. **Journal of Neuroscience**, Society for Neuroscience, v. 38, n. 33, p. 7193–7200, 2018. ISSN 0270-6474. Available at: <https://www.jneurosci.org/content/38/33/7193>. Cited on page 44.

DAUB, C. T. The effect of doors on latent learning. **Journal of Comparative Psychology**, Williams & Wilkins Company, US, v. 15, p. 49–58, 1933. ISSN 0093-4127(Print). Available at: <https://doi.org/10.1037/h0070697>. Cited 4 times on pages 45, 72, 83, and 96.

VORHEES, C. V.; WILLIAMS, M. T. Morris water maze: procedures for assessing spatial and related forms of learning and memory. **Nat Protoc**, England, v. 1, n. 2, p. 848–858, 2006. Cited on page 46.

BESL, P. J.; MCKAY, N. D. Method for registration of 3-D shapes. In: SCHENKER, P. S. (Ed.). **Sensor Fusion IV: Control Paradigms and Data Structures**. [S.l.], 1992. v. 1611, p. 586 – 606. Cited on page 54.

GEIGER, A.; LENZ, P.; URTASUN, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In: IEEE. **2012 IEEE conference on computer vision and pattern recognition**. [S.l.], 2012. p. 3354–3361. Cited on page 54.

SMITH, M.; BALDWIN, I.; CHURCHILL, W.; PAUL, R.; NEWMAN, P. The new college vision and laser data set. **The International Journal of Robotics Research**, SAGE Publications Sage UK: London, England, v. 28, n. 5, p. 595–599, 2009. Cited on page 57.

MUñOZ, M. E. de S.; MENEZES, M. C.; FREITAS, E. Pignaton de; CHENG, S.; RIBEIRO, P. R. de A.; NETO, A. de A.; OLIVEIRA, A. C. Muniz de. xratslam: An extensible ratslam computational framework. **Sensors**, v. 22, n. 21, 2022. ISSN 1424-8220. Available at: <https://www.mdpi.com/1424-8220/22/21/8305>. Cited on page 61.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: TEH, Y. W.; TITTERINGTON, M. (Ed.). **Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics**. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010. (Proceedings of Machine Learning Research, v. 9), p. 249–256. Available at: <https://proceedings.mlr.press/v9/glorot10a.html>. Cited on page 80.