TUM

TECHNISCHE UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK

Biologically inspired spatial navigationusing vector-based and topology-basedpath planning

Tim Engelmann

TUM-I2293

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY MUNICH

Bachelor's Thesis in Engineering Science

Biologically inspired spatial navigation using vector-based and topology-based path planning

Tim Engelmann

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY MUNICH

Bachelor's Thesis in Engineering Science

Biologically inspired spatial navigation using vector-based and topology-based path planning

Biologisch inspirierte räumliche Navigation mit Hilfe von vektor-basierter und topologie-basierter Pfadplannung

Author:	Tim Engelmann
Supervisor:	Prof. Dr.-Ing. habil. Alois Knoll
Advisor:	Dr. rer. nat. Zhenshan Bing
Submission Date:	13.08.2021

I confirm that this bachelor's thesis in engineering science is my own work and I have documented all sources and material used.

Munich, 13.08.2021 Tim Engelmann

Abstract

A combination of different brain cell types allows rodents to navigate in complex environments. These cell types include grid cells that fire periodically across space and keep track of the rodent's movement through path integration. Place cells, in contrast, fire only at a specific location in space. By linking place cells to cells in the prefrontal cortex, the rodent can create a cognitive map of the environment. Inspired by these cell types, researchers have proposed navigational systems for robotic applications. They can be clustered into two main approaches, vector-based and topology-based navigation. During vector-based navigation, the agent heads to the goal directly and exploits shortcuts. By contrast, during topology-based navigation, the agent uses previously gathered information about the environment to avoid obstacles. A combination of the two navigational approaches has proven to unite both of their advantages. However, previous work partly lacks biological plausibility and showed further improvement potential. Here we propose a biologically inspired system architecture, which combines both vector-based and topology-based navigation and can be used for mobile robotic platforms. For performance evaluation, we used a virtual environment and considered five different configurations of a maze. Within that maze, the agent reliably returned to a goal location it had found before. Our work helps to understand how the previously discovered brain cell types are likely to interact and raises the question of how environment-specific information is stored in the prefrontal cortex. Moreover, our proposed system represents a potential solution to the problem of simultaneous mapping and localization (SLAM).

Contents

Abstract **iii**

1. Introduction **1**

2. Background **3**
2.1. Grid Cells . . . 3
2.1.1. Grid Cell Modules . . . 3
2.1.2. Nested Approach . . . 5
2.1.3. Combinatorial Approach . . . 5
2.2. Place Cells . . . 7
2.3. Other Cell Types . . . 8
2.3.1. Head direction cells . . . 8
2.3.2. Border cells . . . 8
2.3.3. Speed cells . . . 8
2.3.4. Prefrontal cortex cells . . . 9

3. Related Work **10**
3.1. Vector-based Navigation . . . 10
3.1.1. Direct Decoding . . . 10
3.1.2. Directed Search . . . 13
3.2. Topology-based Navigation . . . 13
3.2.1. Building up a cognitive map . . . 14
3.2.2. Navigating with the cognitive map . . . 14
3.3. Combining vector-based and topology-based navigation . . . 15

4. Problem Statement **16**

5. Methodology **17**
5.1. Overview of core components . . . 17
5.2. Exploration Phase . . . 18
5.2.1. Grid Cell Model . . . 19
5.2.2. Place Cell Model . . . 22

5.2.3. Cognitive Map . . . 22
5.3. Navigation Phase . . . 24
5.3.1. Vector-based Navigation . . . 26
5.3.2. Topology-based Navigation . . . 30

6. Experiments **32**
6.1. Setup . . . 32
6.2. Grid Cell Modules and Path Integration . . . 33
6.3. Grid Cell Decoding . . . 34
6.3.1. Linear lookahead in two dimensions . . . 35
6.3.2. Phase Offset Detectors . . . 36
6.3.3. Spike Detection . . . 38
6.3.4. Comparison . . . 39
6.4. Cognitive Map . . . 40
6.5. Sub-Goal localization . . . 43
6.6. Overall performance . . . 44

7. Discussion **48**
7.1. Robotic applicability . . . 48
7.2. Biological plausibility . . . 49

8. Conclusion and Future Work **50**
8.1. Conclusion . . . 50
8.2. Future Work . . . 51

A. Supplementary Figures **53**

List of Figures **57**

List of Tables **62**

Bibliography **63**

1. Introduction

Rodents are capable of solving complex navigational tasks, such as finding previously visited locations quickly again. Researchers gathered behavioral data and neurological recordings of rodents and found several distinct brain cell types active during navigation (E. I. Moser, M. B. Moser, et al. 2017). Grid cells fire in a hexagonal pattern at several locations in space (E. I. Moser, Kropff, et al. 2008). They provide the rodent with an internal coordinate system and keep track of its movement through path integration (Edvardsen 2017). Place cells, by contrast, fire only at one specific location in space, allowing the rodent to recognize previously visited places (O'Keefe and Dostrovsky 1971). These cells are located in the hippocampus, which strongly interacts with the prefrontal cortex. Prefrontal cortex cells can be used for memory encoding and storing relevant information about the environment (Eichenbaum 2017). These cell types together allow the rodent to create a cognitive map of the environment, a concept first proposed by Tolman (1948).

Mathematical models can recreate biological cell behavior. Grid cells, for instance, can be represented by continuous attractor networks. In these networks, hexagonal spiking patterns develop and respond to the agent's movement. (E. I. Moser, M. B. Moser, et al. 2017) Researchers have started to develop navigational systems based on these cell models (Edvardsen 2015, Bush et al. 2015, Banino et al. 2018, Erdem and Hasselmo 2012, Edvardsen et al. 2020). These systems are designed to solve navigational tasks consisting of two phases: exploration and navigation. In the exploration phase, the agent discovers the environment, creates a cognitive map, and eventually, by chance, finds the preset goal location. In the navigation phase, the agent must find that goal location again but is expected to do so much faster than during the exploration phase.

There are two main approaches for finding the goal, which can be described as vector-based and topology-based navigation. Vector-based navigation guides the agent along a straight path directly to the goal. This approach is highly effective in environments without obstacles (Edvardsen et al. 2020). Edvardsen (2015), Banino et al. (2018) and Bush et al. (2015) all propose vector-based navigational systems, which are also biologically plausible. However, topology-based navigation outperforms vector-based navigation in environments with densely distributed obstacles. This is because the agent can use the information stored in the cognitive map and follow a previously taken path around an

obstacle (Edvardsen et al. 2020, Erdem and Hasselmo 2012). Edvardsen et al. (2020) demonstrated a system combining both vector-based and topology-based navigation. The system performs very well in cluttered environments yet can also exploit shortcuts and find novel routes. However, given that the implementation of the cognitive map is graph-based rather than based on neurological circuitry, biological plausibility is not fully achieved. Furthermore, we identified improvement potential in the system's grid-cell decoding mechanism and approach of using the cognitive map.

Here we propose an alternative biologically inspired navigational system combining vector-based and topology-based navigation. We based our grid cell model on Edvardsen (2015) and compared three different grid-cell decoding mechanisms for determining the goal vector. The linear lookahead mechanism inspired by Bush et al. (2015) proved to be highly accurate even across long distances. During a linear lookahead, the agent simulates traveling along a straight path without actually moving until its simulated grid cell behavior matches the grid cell behavior experienced at the goal location. Our implementation of a cognitive map solely depends on neurological circuits and was first proposed by Erdem and Hasselmo (2012). The work from Edvardsen et al. (2020) inspired us on how to combine vector-based with topology-based navigation.

We evaluated the system's overall performance in five different configurations of a maze suggested by Banino et al. (2018). The agent returned to the goal location in all configurations, except when we blocked a previously accessible path. Moreover, the agent was able to make use of most shortcuts once they became available.

Our work helps to understand how previously discovered cell types are likely to interact. By connecting their biological behavior to an actual navigational function, we start to grasp their importance in a broader context. Our work also highlights where biological evidence is still lacking. Especially the process of storing environment-specific information in the prefrontal cortex and recalling it for later use requires further research. However, if we do manage to understand the navigational system of the rodent fully, it will be an important stepping stone in brain research, opening doors to other research topics. Moreover, replicating the rodents' navigational capabilities would represent a promising approach to solving the challenge of simultaneous localization and mapping (SLAM).

2. Background

In this chapter, we will describe the fundamental cell types relevant for biologically plausible navigation. Here we will focus mainly on the biological background, while in Chapter 5, we will introduce computational models of replicating the biological behavior.

2.1. Grid Cells

Moser and Moser discovered cells in the medial entorhinal cortex (MEC) of the rat brain that would fire at several equally-spaced locations in the environment (E. I. Moser, Kropff, et al. 2008, Sargolini et al. 2006, Hafting et al. 2005). Those cells, namely grid cells, form a universally applicable spatial coordinate system (Sargolini et al. 2006). Landmarks or well-known locations (e.g., the home base) function as the origin of this coordinate system and allow the rat to position itself in the environment (Tejera et al. 2018). Grid cells also enable path integration during rat movement, keeping track of the rodent's position as it traverses the environment (McNaughton et al. 2006). Researchers sometimes refer to this navigational key concept as "neural odometry" (Tejera et al. 2018).

2.1.1. Grid Cell Modules

As a rodent traverses a linear path, as shown in Figure 2.1, an individual grid cell will fire at equally spaced locations. These locations are a specific distance apart, the grid-scale, and are independent of the speed or path that the rodent chooses (Hafting et al. 2005). To understand how grid cells form a coordinate system, let us first consider a single grid cell. We denote its grid-scale with λ and propose the grid cell fires at the position x_0 where the rodent begins its line traversal. Whenever the cell fires, we know the rodent must be roughly at a multiple of the grid-scale, so $x = x_0 + k \cdot \lambda$, where $k \in \mathbb{N}$. This property splits the traversed line into units of grid-scale size. If the grid cell fires, we know the rodent is positioned at the beginning of a line-unit. Next, we combine several grid cells with the same scale but shift their firing peaks by an offset of ϕ. If we choose ϕ small enough, there will always be a grid cell firing. When the n-th grid cell fires, the rodents position can be described as $x = x_0 + n \cdot \phi + k \cdot \lambda$. Therefore, based on which

grid cell fires we can determine the rodents relative position within a line unit. (Fiete et al. 2008)

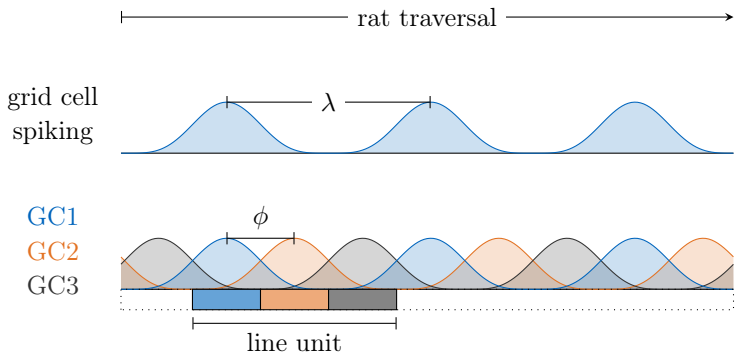


Figure 2.1.: As the rat traverses a linear path, a grid cell will spike every time after traveling a distance of λ. By combining several grid cells (GC) with the same grid-scale λ but a phase offset of ϕ, the space is discretized.

In two dimensions, a grid cell fires in a hexagonal pattern, spanning across the entire environment, as it can be seen in Figure 2.2. Similar to the one-dimensional case, a set of grid cells with the same grid-scale and orientation but different phase offsets form a coordinate system (E. I. Moser, Kropff, et al. 2008, Hafting et al. 2005).

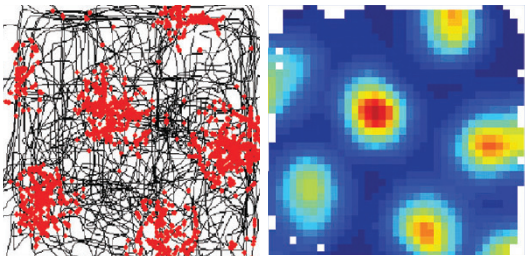


Figure 2.2.: Firing of a single grid cell from the entorhinal cortex of the rat brain. Left, the traveling trajectory of the rat is shown in black, spike locations in red. Right, the cell activity is shown as heat map, where red represents high activity and blue low activity. Figure adapted from E. I. Moser, Kropff, et al. (2008).

The rodent's position is unambiguous within a unit tile, which size is determined by the grid-scale. However, due to the periodic nature of the grid, the rodent does not know in which unit tile it is currently located. (Fiete et al. 2008)

Despite this constraint, recordings of grid cells suggest that closely located grid cells show the same grid-scale and orientation and form a functional independent module. However, the recordings also suggest that several differently scaled modules exist within the layers of the MEC. (Stensola et al. 2012) In sections 2.1.2 and 2.1.3, we will elaborate on two different approaches of combining modules of grid cells to increase the environment size in which the rodent can position itself.

2.1.2. Nested Approach

The "nested approach" of combining grid cell modules as shown in Figure 2.3a and introduced by Edvardsen (2017) ensures that grid cell codes remain unique by relying on modules with large grid-scales. The scale of at least one module exceeds the environment size. This implies that only one unit tile of that grid cell module is considered. By adding modules with smaller grid-scales, the grid becomes more nuanced and, therefore, the localization more precise. The module with the smallest grid-scale determines the resolution of the grid. (Edvardsen 2017)

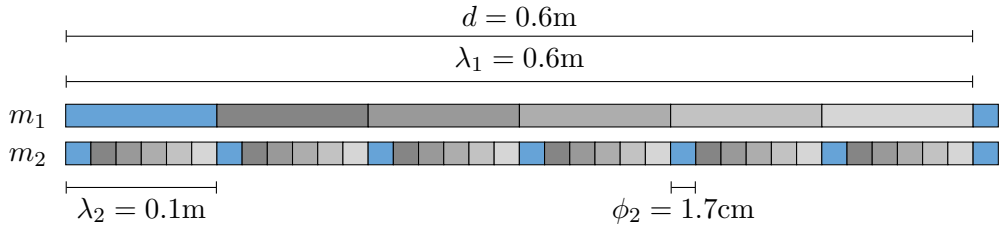
The neurological recordings of Stensola et al. (2012) suggest that the scales of grid cell modules increase linearly with a factor of around 1.42 along the enthornial dorsoventral axis. Edvardsen (2017) embraces that characteristic and increases the coverable environment size by iteratively adding grid cell modules with larger scales. However, Fiete et al. (2008) extrapolated the recordings from Hafting et al. (2005) and suggests as largest grid-scale 1m (linear extrapolation) or 10m (exponential extrapolation). These grid-scales would not be sufficient to cover large scale environments according to the nested approach.

This of course questions the biological plausibility of the nested approach. Nevertheless, it remains worth to be considered, given that it is easily implementable and does not require a highly precise grid cell readout to deliver useful results. This is because each module merely refines the estimate of the larger scale modules further. With the "combinatorial approach" which we will explain next, a slight error in the grid cell readout could lead to a much bigger error in the estimate of the rodents position. (Edvardsen 2017)

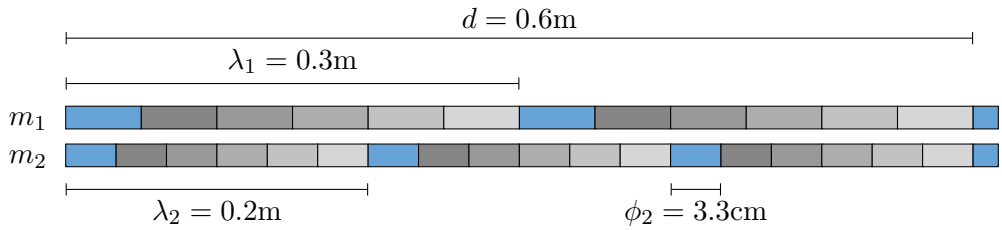
2.1.3. Combinatorial Approach

The "combinatorial approach" as used by Bush et al. (2015), introduced by Fiete et al. (2008) and shown in Figure 2.3b combines several grid cell modules with different but rather smaller grid-scales. While a position defined by an individual grid cell module is ambiguous in larger environments, the combination of several differently scaled modules represents a compact combinatorial code. This code remains unique within an environment size defined by the smallest common multiple of the grid-scales. (Edvardsen 2017)

Adding modules to the system increases the coverable area exponentially (Fiete et al. 2008). For example, a combination of 10 modules with grid-scales $s \in [0.25\text{m}, 5\text{m}]$ is capable of representing an environment of about 3.3km uniquely (Bush et al. 2015,). Given the neurological data, this approach is highly biologically viable. However, as previously mentioned imprecise grid cell readout can quickly lead to large errors in the position estimate. (Edvardsen 2017)



(a) Nested approach



(b) Combinatorial approach

Figure 2.3.: Discretization of the one dimensional space by a combination of two grid cell modules, m_1 and m_2. The colors indicate which of the six grid cells within each module is most active at the respective location. In this example, both approaches discretize the axis for a distance of d unambiguously. (a) m_1 has a relatively large grid-scale and defines the environment size. m_2 discretizes the space further and increases the resolution. Note that usually grid-scales would differ only by factor of about 1.4, but this combination highlights the difference to the combinatorial approach better. (b) The combination of the two grid cell modules forms a grid code. The code is unique across a distance defined by the common multiple of the grid-scales and the number of grid cells per module. Adding another module would greatly increase the coverable environment size.

2.2. Place Cells

O'Keefe and Dostrovsky (1971) were the first to discover a correlation between cell firing in the rat's hippocampus and the current location of the rat. O'Keefe investigated the properties of those cells further and named them "place cells" (O'keefe and Conway 1978. Place cell firing is especially strong in a particular area of the environment called a place field. In contrast, to grid cells that fire at several locations in the environment, a place cell only fires within a singular place field. However, they can be reused across different environments. Place cells form the basis of mapping the environment to a cognitive map, as first introduced by Tolman (1948). (E. I. Moser, Kropff, et al. 2008)

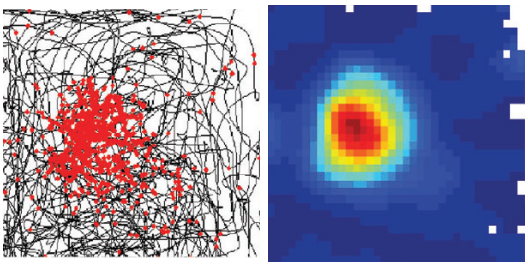


Figure 2.4.: Firing of a single place cell from the rat brain. Left, the traveling trajectory of the rat is shown in black, spike locations in red. Right, the cell activity is shown as heat map, where red represents high activity and blue low activity. Figure adapted from E. I. Moser, Kropff, et al. (2008).

Place cell formation. Grid cells are believed to be the primary input for place cell formation. Some researchers (McNaughton et al. 2006, Solstad, E. I. Moser, et al. 2006) proposed that a Fourier-like sum of the weighted output of grid cells modules with different scales represents the input to the place cell network. Others suggest that "place fields might be generated from any weak spatial input, so long as the hippocampal circuit contains mechanisms for amplifying a subset of these inputs (E. I. Moser, M. B. Moser, et al. 2017)." However, further research on young rats, where grid cells have not developed yet, implies that other cell types are also involved. Border cells, described in section 2.3.2 seem to be of relevance, given that place cells form especially in proximity to boundaries. Nevertheless, the details of place cell formation remain unclear. (E. I. Moser, M. B. Moser, et al. 2017, M.-b. Moser et al. 2015)

2.3. Other Cell Types

Several other cell types are contributing to the navigation system. We will introduce them given their relevance and as they are mentioned repeatedly in related research. However, to reduce complexity, the system introduced in Chapter 5 will not make use of head direction cells, border cells, or speed cells but will retrieve the corresponding information from sensory input. In future work, the system architecture could be extended to implement those cell types as well.

2.3.1. Head direction cells

Head direction cells, recorded in cortical and subcortical regions of the rat brain, fire more strongly in a particular preferred head direction. The preferred directions of the cells are distributed equally in the full circle. The head direction cells integrate the rodent's angular velocity and allow the rat to differentiate between egocentric and allocentric heading direction, an essential prerequisite for vector-based navigation. (Taube et al. 1990a, Taube et al. 1990b, E. I. Moser, M. B. Moser, et al. 2017)

2.3.2. Border cells

Border cells fire along geometric boundaries of the environment, such as obstacles or walls. They, too, have a preferred direction, firing only along specifically orientated obstacles (e.g., the left wall of a room). Border cells were found in the hippocampus and the subiculum. Their influence on place cells remains unclear. Border cells might also enable recalibration of grid cells, improving their path-integration performance over longer distances. (Solstad, Boccara, et al. 2008, E. I. Moser, M. B. Moser, et al. 2017)

2.3.3. Speed cells

Speed cells, discovered more recently, describe a cell population dedicated to encoding the rat's current running speed. Their firing is linearly related to the running speed, and a small number of cells can already represent highly accurate values. Speed cells connect tightly to grid cells and can serve as an input to enable path integration. (Kropff et al. 2015, E. I. Moser, M. B. Moser, et al. 2017)

2.3.4. Prefrontal cortex cells

The prefrontal cortex is closely connected to the hippocampus, and together they are believed to be capable of "memory encoding and context-dependent memory retrieval (Eichenbaum 2017)." Storing information about the environment in the form of a "cognitive map" is necessary to perform topology-based navigation. Erdem and Hasselmo (2012) propose a navigational model with a variety of cell types located in the prefrontal cortex, namely recency cells, topology cells, and reward cells. While the existence of these particular cell types lacks biological evidence, the recordings from Place et al. (2016), Burton et al. (2009) and Hok et al. 2005 imply that reward information could very well be stored in the medial prefrontal cortex and retrieved for goal-driven navigational tasks.

3. Related Work

In this chapter, we will present the concepts and papers relevant to the navigational system which we will propose in Chapter 5. We assigned the related work to one of three categories: systems relying primarily on vector-based navigation, systems performing topology-based navigation and systems combining both navigational approaches. We will dedicate a section to each of these categories.

3.1. Vector-based Navigation

During vector-based navigation, an agent calculates the vector between the current and the goal position. It then follows that navigational vector and takes the most direct path to the goal. Vector-based navigation performs especially well in environments without obstacles, as the direct path is also the shortest. (Edvardsen et al. 2020)

Grid cells can encode the agent's position uniquely in large-scale environments as described in section 2.1. Therefore, we can determine the goal vector by comparing two sets of grid cell firing, the one at the current and the one at the goal position. Several decoding mechanisms exist to perform that comparison. Bush et al. (2015) proposed to cluster the mechanisms in "direct decoding" and "directed search." We will provide a description and examples for both clusters in the following sections 3.1.1 and 3.1.2.

3.1.1. Direct Decoding

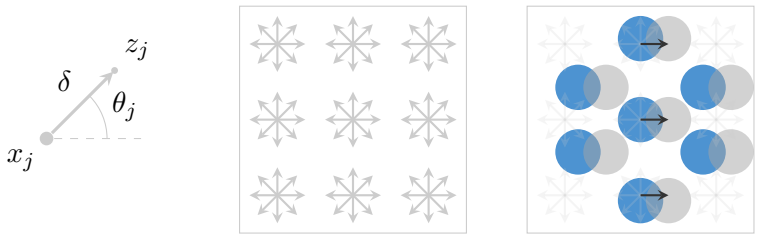
Mechanisms that fall into the "direct decoding" cluster rely on an additional neural circuit to decode the vector between two grid cell patterns. The grid cell spikings at the current and the one at the goal position represent the input to that circuit. The output is the 2D vector between these two positions which the agent can follow to reach the goal. The advantage of these mechanisms is their low computational effort, allowing for very frequent vector calculations. The disadvantage is that they require additional system architecture consisting of dedicated decoder neurons, for which limited biological evidence exists. (Bush et al. 2015, Poulter et al. 2018)

In the following, we will touch upon two direct decoding mechanisms more closely, as we use them as reference for performance evaluation in Chapter 6.

3.1.1.1. Phase-offset detectors

Edvardsen (2015) proposed such a direct decoding mechanism, which includes a network of so-called "phase-offset detector" neurons. As input, it considers two-dimensional grid cell sheets, implementing a continuous attractor model. The hexagonal spiking pattern shifts on these sheets according to the movement of the agent. For details on this grid cell model, refer to section 5.2.1.

A phase-offset detector is a neuron tuned to detect a shift/offset of δ between a grid cell module's current and goal spiking in a specific direction θ_j. It is associated with a fixed location x_j on the neuron sheet. The phase-offset fires most strongly if at x_j there is a peak in the current grid cell pattern $\mathbf{s}$ and a peak at the offset position $z_j = x_j + \delta\hat{e}_{\theta_j}$ in the goal grid cell pattern $\mathbf{t}$. To ensure that the mechanism detects all possible offsets, it uses a network of phase-offset detectors, spread out across the neuron sheet and tuned for different equally spaced directions. In Figure 3.1 we show an exemplary network of such phase-offset detectors and visualize their behavior. Edvardsen proposed to add a motor neuron to the network for each direction and to connect it to all phase-offset detectors tuned for the same characteristic direction. The goal vector then computes as the vectorial mean of the motor neuron firing. (Edvardsen 2015)



(a) Single detector (b) Network of detectors (c) GC offset detection

Figure 3.1.: (a) Detector neuron located at x_j and tuned to detect phase-offsets in the direction of θ_j. It is excited most strongly at a pattern offset of δ. (b) Exemplary network of $3 \times 3 \times 8$ phase-offset detectors, equally spaced across a neuron sheet. Here we distinguish between 8 offset directions only. (c) Current grid cell spiking $\mathbf{s}$ in blue and goal spiking $\mathbf{t}$ in grey. Only three phase-offset detectors spike significantly, as their location x_j is favorable and they are tuned for $\theta_j = 0°$.

3.1.1.2. Long short-term memory networks

Banino et al. (2018) took a different approach to perform vector-based navigation. Instead of trying to recreate the functionality of neurons like grid cells directly, they trained recurrent networks on navigational input data of simulated rodent movement. During training, they compared the estimated position and head direction with the actual data in each timestep. As recurrent networks, they used long short-term memory (LSTM) networks, and as input data, the velocity vector of the agent and occasionally some visual input. In these networks, grid cell-like behavior emerged, which could be used to perform path integration. The output of the network represented a grid code uniquely encoding the current position of the agent.

After establishing the grid cell-like network, they trained a policy LSTM to enable the agent to navigate vector-based to the goal. As input, it received the current grid code, the goal grid code, reward info, the previous action taken, and occasionally preprocessed visual input. As output, it returned a heading speed and direction to navigate to the goal. For a sketch of the overall architecture, refer to Figure 3.2. The proposed system proved to be a highly performing vector-based approach for biologically inspired navigation. (Banino et al. 2018)

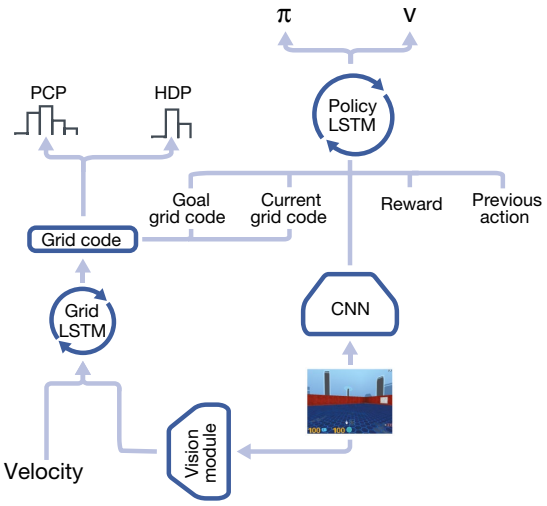


Figure 3.2.: Architecture of the system proposed by Banino et al. (2018). It uses a LSTM to simulate grid cell like behavior and another LSTM to decode two grid cell codes into an actionable goal vector. Figure from Banino et al. 2018

3.1.2. Directed Search

Mechanisms of the "directed search" cluster start their goal vector search either at the goal location or the current location. Then a constant velocity vector in several search directions "decoupled from the animal's actual motion (Bush et al. 2015)" is applied to the corresponding grid cell spiking. If a search direction matches the goal vector direction, both grid cell spikings will become more and more alike until they eventually match. However, if we continue to apply the velocity vector, the grid cell spikings will drift apart again. In that case we surpassed the goal in our lookahead. The time where both grid cell spikings match best is denoted as critical search time. Following this approach, the relative offset and goal vector of the two grid cell spikings can be determined. Such grid cell decoding mechanisms do not require additional system architecture. This is because it only relies on the grid cell network, which is in anyway necessary to perform path integration. However, vector computation takes significantly longer than with a direct decoding mechanism. Computation time scales with the goal vector length. (Bush et al. 2015, Poulter et al. 2018)

Biological recordings from Spiers et al. (2018) suggest that in place cells activity increases with the distance to the goal. As grid cells and place cells are closely linked, this could further support the biological viability of the directed search mechanism.

One open question remaining is how the search directions are being chosen? Erdem and Hasselmo (2012) started the search from the current position. The agent then looked ahead in several equally spaced directions. The higher the number of directions, the more likely it was that a look-ahead will find the goal vector. Bush et al. (2015) proposed to perform a search along two linearly independent axes. Instead of comparing the entire grid cell spiking at once, they considered the spiking vector projected on the respective axis. The overall goal vector was then a linear combination of the two vectors found during the searches.

3.2. Topology-based Navigation

During topology-based navigation, an agent relies on its knowledge of the environment to find a path to the goal. This approach performs especially well if obstacles block the direct path and the agent has to find a way around. The two main aspects of topology-based navigation are: how can we store information about the environment in a cognitive map, and how can we use that map later to find the shortest path to the goal? (Edvardsen et al. 2020)

3.2.1. Building up a cognitive map

Edvardsen et al. (2020) used a graph-based approach to build a cognitive map. Whenever the agent visited a location it has not been before, it created a new graph node corresponding to a place cell in the biological context. At the moment of creation it saved a snapshot of the current grid cell spiking and linked it to the graph node. When the agent visited the locations of two graph nodes consecutively, the nodes were linked bidirectionally. If the agent found the goal, the corresponding node was marked accordingly. With the extensive exploration of the environment, this graph became more and more detailed and represented the cognitive map. While this approach successfully demonstrated the different steps involved in creating a cognitive map, Edvardsen et al. (2020) decided on a graph-based implementation "for simplicity" reasons. It can be argued, that in this aspect it lacks biological plausibility, given that it does not rely on neurological circuitry only.

Erdem and Hasselmo (2012) on the other hand proposed a functionally quite similar approach as Edvardsen et al. (2020) yet solely used a network of neurons and made use of prefrontal cortex cells. As for Edvardsen et al. (2020), at new locations, place cells were created but then actually linked to the grid cell network. Furthermore, a network of prefrontal cortex cells kept track of the proximity of place fields. Reward information was saved in dedicated cells, which imposed a reward gradient towards the actual goal. For a detailed description, refer to section 5.3.2. Although no recordings of cells with these exact properties exist, the prefrontal cortex does play a role during navigation as described in section 2.3.4, making this approach biologically viable.

3.2.2. Navigating with the cognitive map

Edvardsen et al. (2020) relied on the cognitive map when the agent faced an obstacle. The agent could compute the goal vector between the current and goal locations using the saved grid cell codes. When the goal vector was blocked, the agent had to identify a subgoal. It started with computing the goal vectors to graph nodes directly connected to the goal node and continued with nodes further away until a non-blocked vector is found. That node represented a sub-goal that the agent could travel to and from which it was sure that a valid path to the actual goal existed. Erdem and Hasselmo (2012) chose to perform a forward look-ahead instead. The agent pre-played several non-blocked traveling directions from the current positions, as described in section 3.1.2 and 5.3.2. It then chose the direction that encountered a place cell with the highest associated reward and was, therefore, closest to the actual goal.

3.3. Combining vector-based and topology-based navigation

Edvardsen et al. (2020) combined the vector-based with the topology-based navigational approach by switching between them on demand. By default, the agent navigated vector-driven. If the agent approached an obstacle at a slanted angle, it adapted the navigational vector until successfully avoiding the obstacle. However, if the agent encountered an obstacle head on it switched to topology-based navigation. It found a suitable path around the obstacle and followed that path at first, as described in section 3.2.2. When the agent was clear of the obstacle, it would switch back to vector-based navigation. Otherwise, it continued to follow the known path until the goal is reached. The proposed system was able to exploit shortcuts but also effectively navigated around obstacles. Combining vector-based with topology-based navigation therefore seems to be the most promising approach. However, as described in section 3.2.1, Edvardsen et al. (2020) decided for a graph-based implementation of the cognitive map, which reduces the biological plausibility strongly.

4. Problem Statement

This thesis aims to develop a biologically inspired system, enabling an artificial agent to navigate in its environment. The core components of the proposed system should thereby rely on neurological circuitry only, representing the cell types found in the rodent's brain. The agent has to remember the location of a single goal, which it finds during the exploration phase. It is asked to return to that specific goal during the navigation phase. The environment can hold several obstacles, limiting the number of valid paths. Yet, the proposed system should aim to find the shortest path in a variety of situations. If the chosen path is short, the system is considered to be efficient. Above efficiency, the system should be reliable, meaning that the agent reaches the goal eventually during the navigation phase. Given the navigational capabilities of rodents, we expect that the proposed system will combine both vector-based and topology-based navigation.

In summary, we aim to develop a system that solves a goal-driven navigational task while being biologically plausible, efficient, and reliable.

5. Methodology

In this chapter, we will introduce the proposed algorithm, combing vector-based with topology-based navigation. First, we will give an overview of the proposed system in section 5.1. We will then provide more details on its functionality during the exploration and navigation phase. In the exploration phase, the agent learns about the environment and creates a cognitive map. In the navigation phase, we then ask the agent to efficiently find its way to the goal using that cognitive map.

The cognitive map relies on several neurological networks embedded in the system architecture. In section 5.2 we will propose an architecture used during the exploration phase and capable of controlling the agent to explore its surroundings. In section 5.3 we will then outline a system that uses the neurological networks trained in the exploration phase and combines vector and topology-based navigation.

5.1. Overview of core components

The neurological networks are the central part of the proposed algorithm and include grid cells, place cells, and several other cell types. Together, these cells form a cognitive map of the environment that allows us to perform vector and topology-based navigation. In the following, we will describe how these cells are connected with each other, as also shown in Figure 5.1. Implementation details are provided in sec 5.2.

While the agent traverses its environment, the agent's speed and traveling direction are given as input to the system. Grid cells allow the agent to perform path integration and encode the current position vector, whereas place cell firing is especially strong at a specific location in the environment. Once the agent visits a place it has not been before, we create a new place cell and connect it to all currently active grid cells. Whenever this specific set of grid cells fire, the place cell will also fire. Remembering a specific location in the environment via a place cell allows us to save more information about this location. Relevant information to keep track of include: if the agent visited the location recently, if it found a reward there, and if we know that other locations are nearby. We encode this information in a cortical column connected to the place cell. The cortical column includes a recency cell (last time of visit), reward cell (reward info), and topology cell

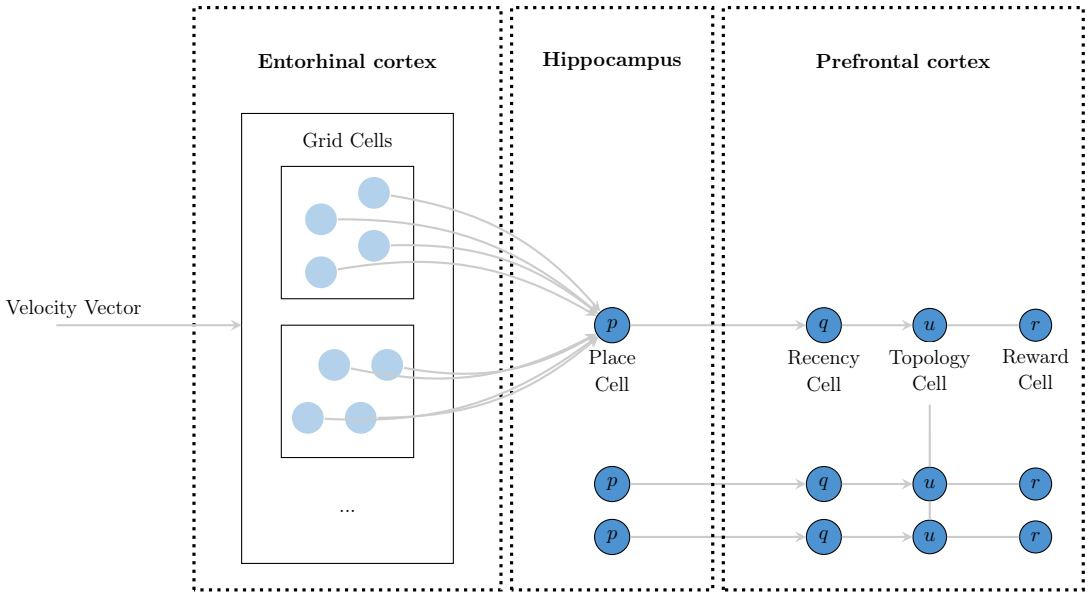


Figure 5.1.: Overview of the central part of the system architecture. Several neurological networks are combined to create a cognitive map of the environment.

(connections between locations). For more details on the biological background of these cells, refer to Chapter 2.

All of these cells together make up the cognitive map. This map becomes more detailed throughout the exploration phase and covers a more significant portion of the environment.

5.2. Exploration Phase

Exploration phase controller. During the exploration phase, a controller determines the agent's current position and computes the vector to the next navigational goal. The goal selection can either be random or follow a fixed hard-coded exploration path. As soon as the agent reaches a navigational goal, it proceeds to the next one. We pass the current goal vector to the Course Adjustment controller.

Course Adjustment controller. The controller is responsible for recognizing obstacles nearby and preventing the agent from colliding. We realized that for the exploration phase, a basic obstacle avoidance approach would be sufficient. The agent checks in 16 allocentric directions between $[0, 2\pi[$ for obstructions. If the minimum distance in any of those directions falls below a threshold of 0.3m, the agent backs up from the

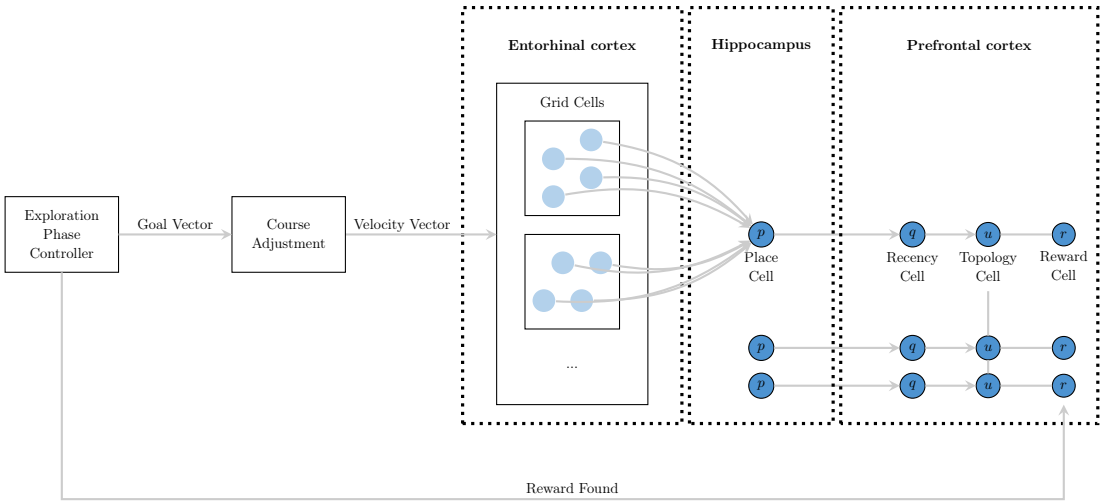


Figure 5.2.: System overview of the exploration phase where the agent starts creating a cognitive map of the environment.

obstacle for 0.5m before choosing the next navigational goal (randomly). Note that for biological plausibility, an obstacle avoidance controller including border cells would be more favorable. Implementing such a system could be the subject of further research. After adapting the goal vector if necessary, we compare it with the current heading direction and calculate the respective motor gains. The agent's corresponding traveling speed and direction represent the velocity vector. Furthermore, we check the agent's current position in the environment for rewards (e.g., a food source or the home base).

The velocity vector and the reward information represent the input to the biologically inspired system, already described in 5.1. In the coming sections, we will further elaborate on the processing of these input signals.

5.2.1. Grid Cell Model

In Chapter 2.1 we described the organization of grid cells in modules and how they can enable path integration. We implemented the grid cell system by closely following Edvardsen 2017 and its underlying work, mainly Edvardsen 2015, as well as Burak and Fiete 2009. The proposed architecture implements a continuous attractor network model (CAN) and proved to enable long-range path integration by combing several modules of grid cells.

5.2.1.1. Functionality of a single Grid Cell Module

One module consists of $n \times n$ (with n = 40) neurons, forming a grid cell sheet. The North (N) and South (S), and the East (E) and West (W) edges of this sheet are connected, which we consider when calculating distances between two points on the sheet. Each neuron is slightly tuned towards one direction, ensuring that a velocity input signal leads to a shift in the spiking patterns. We assign one of the preferred directions $\hat{e}_{\theta_i} \in [W, N, S, E]$ according to the x and y coordinate of the neuron in the sheet. $\hat{e}_{\theta_i}$ is picked based on the idx calculated by $(2 \cdot (y \mod 2) + x \mod 2)$. Following this approach, tiles of four neurons with different preferred directions develop. If no velocity input is applied, the directions cancel each other out, leading to a stable activity pattern.

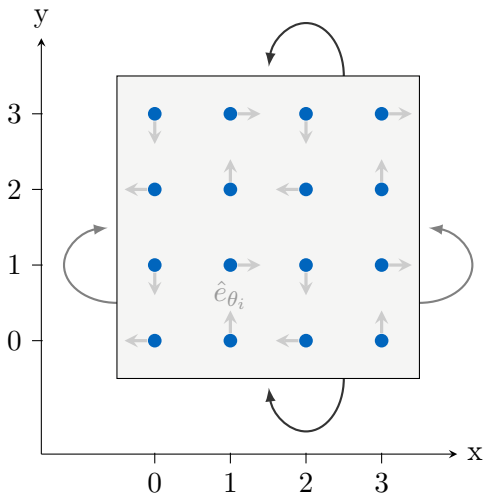


Figure 5.3.: Neuron sheet of $n \times n$ neurons (blue) where $n = 4$. The East and West, as well as the North and South edges are connected. Light gray arrows indicate the preferred direction $\hat{e}_{\theta_i}$ of each neuron, which is used when calculating the recurrent connectivity profile and when applying an external velocity input to the network.

The connection weight w of neuron i to neuron j is calculated by first determining the shortest distance d between the shifted position of $i - \hat{e}_{\theta_i}$ and the actual position of j in the neural sheet. The final weight can the be calculated via the recurrent connectivity profile, given by:

$$rec(d) = e^{-\gamma d^2} - e^{-\beta d^2} \tag{5.1}$$

where $\gamma = 1.05 \cdot \beta$, $\beta = 3/\lambda^2$ and $\lambda = 15$. The overall connectivity weight vector of neuron i is than given by $w^{rec}_{x_i - \hat{e}_{\theta_i}}$ and is of dimension $n^2 \times 1$.

After each time step the spiking values of all grid cells are updated according to the following rule, where s is the grid cell spiking vector:

$$\tau \frac{ds_i}{dt} + s_i = f(s \cdot w^{rec}_{x_i - \hat{e}_{\theta_i}} + B_i) \tag{5.2}$$

with $dt = 10ms, \tau = 100ms, f(x) = max(0, x)$ and

$$B_i = 1 + g_m \alpha \hat{e}_{\theta_i} \cdot v \tag{5.3}$$

with $\alpha = 0.10315$, v the input velocity vector and g_m a module specific scaling factor. $|v| > 0$ leads to shift in the grid cell pattern. In each time step we retrieve the velocity vector of the agent's movement. Then we update the grid cell spiking values, with the equation (5.2.1.1) solved for dsi with an implicit Euler scheme.

5.2.1.2. Combining several Grid Cell Modules

To increase precision and cover larger environments, we combine several Grid Cell modules with different scaling factors g_m according to the nested approach described in 2.1.2. Therefore, the scale of the most extensive grid cell module exceeds the environment size, while the grid cell modules with lower scale increase the precision. In theory, the environment size that the combinatorial approach can cover is much larger than with the nested approach. However, it requires high precision in motion tracking and grid cell readout to determine the position correctly. Otherwise, it is prone to significant errors. Large environment sizes were not a requirement for this thesis, making the nested approach the preferred choice.

The nested approach defines the scaling factors g_m to be in a range of $[g_{min}, g_{max}]$, so that the g_{min} is small enough that grid cell patterns stay unambiguous and g_{max} is large enough to provide sufficient precision. However if g_{max} becomes to large than the velocity input, applied in (5.2.1.1) is scaled too much and the grid cell pattern becomes unstable. If M grid cell modules are used in total, than g_m is given by:

$$g_m = g_{min} \cdot R^m \tag{5.4}$$

where $R = (g_{max}/g_{min})^{1/(M-1)}$ and $m \in [0, M[$. In Edvardsen 2017 the effect of pinning is described. Pinning occurs for grid cell modules with very small g_m because the scaled input velocity $g_m \cdot v$ becomes too small to affect the grid cell patterns. We investigated this effect but realized that our environment size does not exceed a size so that pinning would become relevant. For larger environments (with a radius > 100m), we recommend following the "Poisson-neuron" approach suggested in Edvardsen 2017.

5.2.1.3. Initialization and Simulation

We initialize the grid cell network before the simulation start, with initial spiking values $s_i < 10^{-4}$ and by applying random small velocity vectors for 1000 time steps of $dt = 10$ms. Hexagonal patterns quickly form in each module. For more details, refer to 6.2.

With the grid cell modules initialized, the simulation can start. After each time step of $dt = 10$ms, the velocity vector is applied to the grid cell network, leading to a shift in the hexagonal pattern. The magnitude of the shift depends on the scale factor g_m of each module. With correctly chosen scale factors, each location in the environment corresponds to a specific set of grid cells firing.

5.2.2. Place Cell Model

Place cells are linked to a specific location in the environment as explained in Chapter 2.2. Therefore, we can use them to remember specific points on the agent's traveled path and perform navigation with them later on. The interplay between grid cells and place cells is the topic of many papers as described in 2. For our system, we decided to use a simple one-directed connection model. When the robot traverses a region it has not been before, it creates a new place cell. All grid cells with a spiking value $s_i > 0.1$ form a connection to that place cell with weight $w = 1$, representing a many-to-one mapping. Only the spiking of connected grid cells will propagate to the place cell as the agent continues its path. To calculate the firing value p_i of a particular place cell, we filter the current normalized grid cell spiking $s/|s|$ by its grid cell connectivity profile, denoted by c_i. If there are several grid cell modules, we have to average across them:

$$p_i = \frac{1}{M} \cdot \sum_{m=0}^{M-1} \frac{s_m \cdot c_{m|i}}{|s_m|} \tag{5.5}$$

As the agent moves away from the center of a place field, the place cell firing p_i decreases. A new place cell forms when the highest place cell spiking value falls below the threshold of $p_i < 0.85$ or the agent finds the reward.

5.2.3. Cognitive Map

To allow the agent to remember specific information about the locations it visited, we implemented a cognitive map based on the concept described in Erdem and Hasselmo 2012. We link a cortical column to every newly created place cell. This column includes a recency cell, a topology cell, and a reward cell.

5.2.3.1. Recency Cells

In each time step, we check for place cells that exceed $p_i > 0.85$ and determine the most active one (if there are several). The recency cell in the cortical column linked to that place cell fires with a value of $q_i = 1$, indicating that the agent is currently in this place field. The spiking value q_i of all other recency cells decays over time, according to:

$$q_i = \epsilon^{-\lambda t} \tag{5.6}$$

where we chose $\epsilon = 2$ and $\lambda = 0.01/dt$. Therefore, the recency cells hold the information when the agent visited a specific place field last.

5.2.3.2. Topology Cells

Whenever the agent enters a new/different place field, we can check which place fields it had recently visited. These place fields are likely to be in close spatial proximity, which is relevant information for path planning. Topology cells allow us to store that information. If two place fields are close, we create a bidirectional connection between the two respective topology cells. Upon entry in a new/different place field, the new connections that we can identify T_{new} are expressed by the following:

$$T_{new} = H(q^T - \delta) \cdot H(q - 1) \tag{5.7}$$

where $H()$ is the Heaviside step function with $H(0) = 1$ and $\delta = 0.5$, corresponding to the recency threshold. This formula states, that we crossreference the recently visited cells ($q_i > \delta$) with the currently visited cell ($q_i = 1$). The overall topology cell connection matrix T is then updated as $T = T \vee T_{new} \vee T_{new}^T$.

5.2.3.3. Reward Cells

To perform navigation during the navigation phase, it is essential that the agent finds and remembers a goal during the exploration phase. Once the agent has found a reward location in the environment, we assign a reward value of $r_i = 1$ to the reward cell linked to the respective place field.

To allow the agent to find the reward through topology-based navigation at a later point again, we additionally induce a reward gradient, increasing towards the actual goal. Erdem and Hasselmo 2012 refers to this concept as a diffusive reward. To implement the diffuse reward, we make use of the network of topology cells. Whenever T changes, we also update the reward cells. We set the reward cell of the actual goal to one and all

others to zero. Then we let the reward propagate through the network up to the nth neighborhood. Reward Cells representing the kth neighbourhood ($k < n$) will then have a reward value of $r_i = 1/(k+1)$. In practice, this reward propagation follows the equation:

$$r_i^{(k)} = \max\{\frac{1}{k+1} \cdot H(r^{(k-1)} \cdot T)_i \ , \ r_i^{(k-1)}\} \tag{5.8}$$

where $H()$ is the Heaviside function with $H(0) = 0$ and k runs from 0 to n.

5.3. Navigation Phase

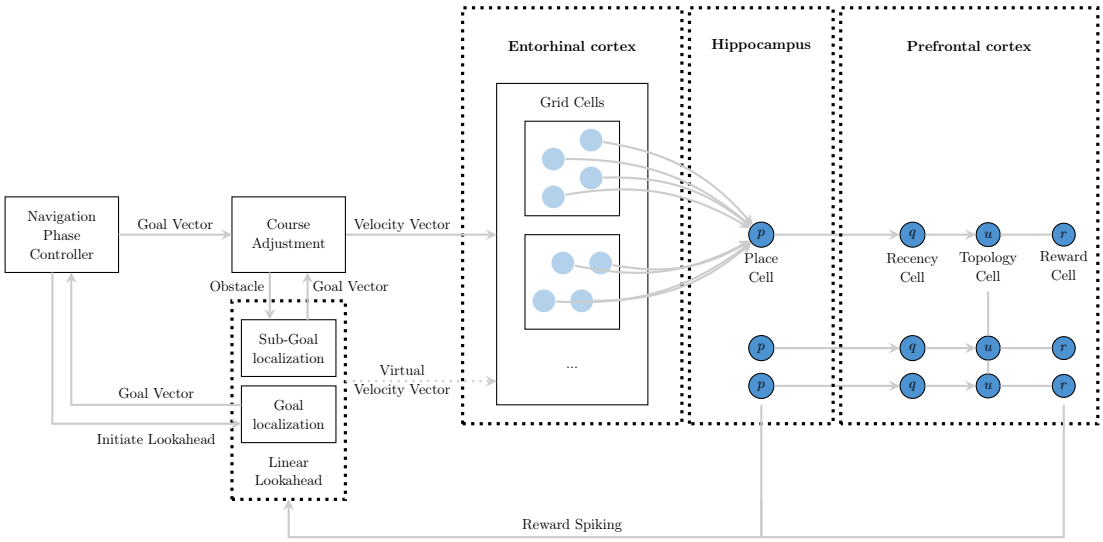


Figure 5.4.: System overview of the Navigation Phase where the agent uses the cognitive map to navigate to the reward location.

When the agent enters the navigation phase, it efficiently returns to the previously found reward location. As described in chapter 3, we can differentiate between a vector-based and topology-based navigational approach. Combining both allows the agent to explore shortcuts and effectively navigate in cluttered or maze-like environments.

Navigation Phase controller. The navigation phase controller determines in what direction the agent should travel. At the beginning of the navigation phase, it initiates a linear lookahead to find the goal vector. During the linear lookahead, we apply a virtual velocity vector to the grid cell modules and pre-play several trajectories until we see the place cell corresponding to the reward location spike. Refer to chapter 3 for more details on the concept and to section 5.3.1 for a description of our implementation. We

recalculate the goal vector during vector-based navigation once the agent has covered 80% of the original vector length by starting a new linear lookahead. If the original goal vector is shorter than $d < 0.3$m, we do not perform any further recalculations. During topology-based navigation, the controller only initiates a new sub-goal localization once the agent reached the previous sub-goal ($d < 0.3$m). For both vector-based and topology-based navigation, if we did not recalculate the goal vector, we update it by subtracting the traveled distance in the last time step.

Course Adjustment controller. The computed goal vector is provided as an input to the course adjustment controller. The controller checks for obstacles in 16 allocentric directions. Similar to the controller in the exploration phase, it initiates a backup maneuver if it detects an obstacle closer than 0.3m away. However, we check additionally for obstacles in the heading and goal vector direction during the navigation phase. If we encounter an obstacle in any of those two directions less than 0.6m away, the agent switches from vector-based to topology-based navigation. As the original path to the goal is blocked, the agent must identify a suitable sub-goal from which the goal is reachable. We can use the information stored in the cognitive to identify such a sub-goal. The agent performs a linear lookahead in several non-blocked directions. We consider a direction θ as blocked if either the distance to the obstacle $d_\theta < 1.3$m, or one of the neighboring directions $d_{\theta\pm\alpha} < 0.9$m. Taking the neighboring directions into account ensures that the agent does not accidentally hit a cornered object. For a visualization, refer to Figure 5.5.

During a linear lookahead in a specific direction, the agent might encounter a place cell and its corresponding reward cell virtually. We keep track of the highest reward spiking, and after performing the lookahead in all chosen directions, we can pick the direction leading to the highest reward spiking. The agent knows that if it travels along that direction, it will eventually get to a location from where it had previously found the goal. For implementation details on the directed linear lookahead, refer to section 5.3.2. During topology-based navigation, the agent constantly checks if a prior blocked traveling direction becomes available. If so, it performs a new sub-goal localization. However, if all traveling directions are unblocked again or the goal is very close, it switches back to vector-based navigation. After the (eventual) adaption of the goal vector, the controller computes the motor gains accordingly. We then pass the corresponding velocity vector to the neurological networks for processing, similar to the exploration phase. Once the goal vector length is below a threshold of $d < 0.1$m, we consider the goal as reached, and the agent halts until the end of the simulation.

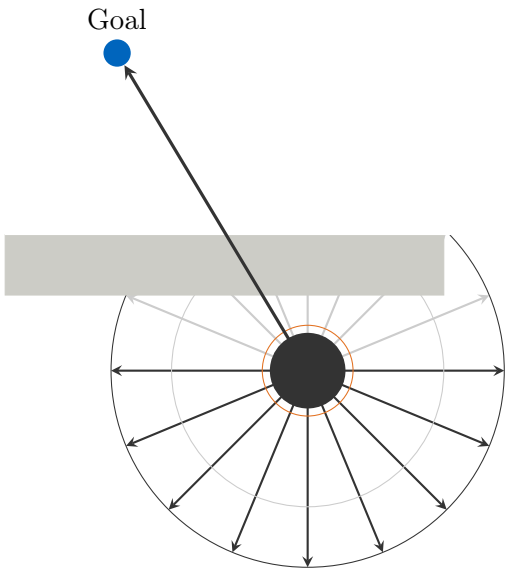


Figure 5.5.: The agent checks for obstacles in 16 allocentric, the goal vector and the heading direction. lookahead directions are considered as blocked if $d_\theta < 1.3$m or $d_{\theta \pm \alpha} < 0.9$m. The thick arrow is the blocked goal vector, light gray arrows represent blocked directions, all other dark grey arrows non-blocked directions. The dark grey circle marks the distance $d = 1.3$m, the light grey circle a distance of $d = 0.9$m and the orange circle the minimum distance of $d = 0.3$m.

5.3.1. Vector-based Navigation

In section 3.1 we outlined several approaches to compute the navigational vector between two sets of grid cell spiking. We implemented and compared three approaches. Refer to 6.3 for an evaluation of their performance. We decided to use the linear lookahead approach, given that it has the highest biological plausibility, highest accuracy and does not require any additional system architecture, as depicted in 3.1.

We are using two different versions of the linear lookahead for goal and sub-goal vector computation. During the goal localization, we are simply looking for the highest reward spiking yet do not know in what direction we should look. In contrast, we are not looking for the overall highest reward spiking during sub-goal localization but for the most promising direction not blocked by an obstacle. Both problem statements can be solved with the linear lookahead yet look a bit different in the implementation. In this section we will focus on the goal vector localization, while in section 5.3.2 we will elaborate on

the sub-goal localization.

To determine the goal vector, we are performing the linear-lookahead separately in the x and y-direction. While this does not pre-play the actual path the agent will travel, it ensures that we will find the goal. We define a lookahead speed $|v|$ depending on the arena size and then the four look-ahead velocity vectors $|v| \cdot [\hat{e}_x, -\hat{e}_x, \hat{e}_y, -\hat{e}_y]$ respectively. For each of the directions, we start a linear-lookahead.

Projected place cell firing value. At the beginning of each time step, we compute the projected place cell firing value. As we are looking for a horizontal/vertical alignment only, we have to project the grid cell pattern on the respective axis and filter it by the projected grid cells to place cell connections. If both of the projections align, we have a high spiking value, otherwise a low one.

The x-directed place cell firing of an individual place cell $p_{i|m|x}$ related to one module can therefore be computed as:

$$p_{i|m|x} = (\sum_{x=0}^{n} \hat{S}_{m|xy} \cdot \sum_{x=0}^{n} C_{i|m|xy})/|\hat{s}_m|^2 \tag{5.9}$$

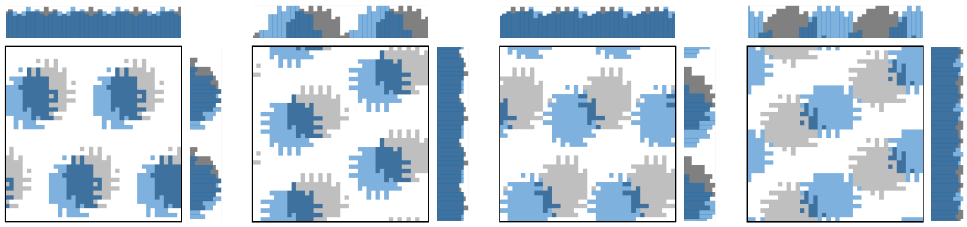
where $\hat{S}_m$ is a $n \times n$ matrix, indicating what grid cells are currently spiking. It takes a value of 1 if the corresponding grid cell j fires ($s_{m|j} > 0.1$) and 0 if not. $C_{i|m}$ is the connectivity profile of place cell i for module m reshaped to the same dimension.
It is important to note that this formula only provides satisfactory results if the grid cell orientation is favorable. The grid cell pattern has to be oriented so that the pattern is still recognizable in the projection. See 5.6 for an example of those different alignments. Therefore, we paid attention to use a combination of grid cell modules tuned to different directions. To calculate the overall x-directed place cell firing of an individual place cell $p_{i|x}$, we averaged the spiking value of all modules being responsive to the x-direction.

Reward spiking values. After computing the directed place cell spiking, we can determine the reward spiking by multiplying the place cell spiking with the reward saved in the corresponding reward cell. We then determine the place cell with the highest reward spiking and determine the x-directed reward spiking:

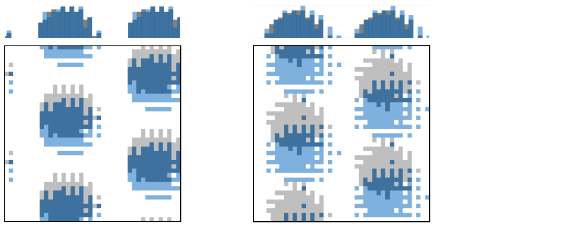
$$b_x = \max\{p_x \cdot r\} \tag{5.10}$$

We overwrite the entry if we are at the start of the simulation or the reward spiking exceeds the previously saved reward for that or the opposing travel direction. The virtual distance traveled after k time steps in the current lookahead direction then corresponds to $d = |v| \cdot dt \cdot k$. If, however, the reward spiking is decreasing and falls below a threshold of $b^k < 0.85 \cdot b_{max}^{0\ldots k-1}$ then the lookahead in that direction is aborted.

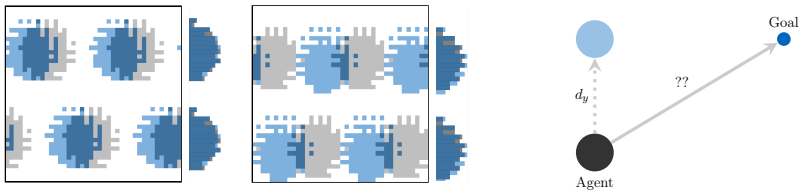
Goal vector computation. As long as we did not reach this threshold or the timeout, the lookahead continues. We apply a virtual velocity vector corresponding to the lookahead direction to the grid cell modules. Then the directed place cell firing is computed again, and so on. We realized that it does not necessarily make sense to evaluate the place cell and reward spiking after every time step $dt = 10$ms. We, therefore, implemented an option to check it only every n^{th} time step to reduce the computational power. Before starting the lookahead in a new direction, we reset the grid cell modules to the actual current spiking.
After the simulation of all four directions, the goal-vector corresponds to $[d_x, d_y]$. d_x and d_y are the distances traveled at the moment of the highest reward spiking in the corresponding direction. The agent can then start to travel in the direction of that goal vector.



(a) Initial state at $k = 0$



(b) State after $k \sim 500$ look-ahead in x-direction



(c) State after $k \sim 260$ look-ahead in y-direction

Figure 5.6.: (a) The current grid cell spiking $\hat{S}$ is represented in blue, and the connections C_i between the grid cells and the target place cell i are represented in grey. We are showing four grid cell modules with different velocity scaling factors g_m. The modules with indexes 0 and 2 can detect changes in the x-direction and modules 1 and 3 in the y-direction. The projections are visualized as bar charts right next to the module in the corresponding direction.
(b) linear lookahead after ~ 500 time steps in the positive x-direction. Modules 1 and 3 are shown at the moment when the patterns align best. The x-directed place cell firing value of an individual place cell $p_{i|x}$ reaches its maximum here.
(c) linear lookahead after ~ 260 time steps in the positive y-direction. Modules 0 and 2 are shown at the moment when the patterns align best. The y-directed place cell firing value of an individual place cell $p_{i|y}$ reaches its maximum here.

5.3.2. Topology-based Navigation

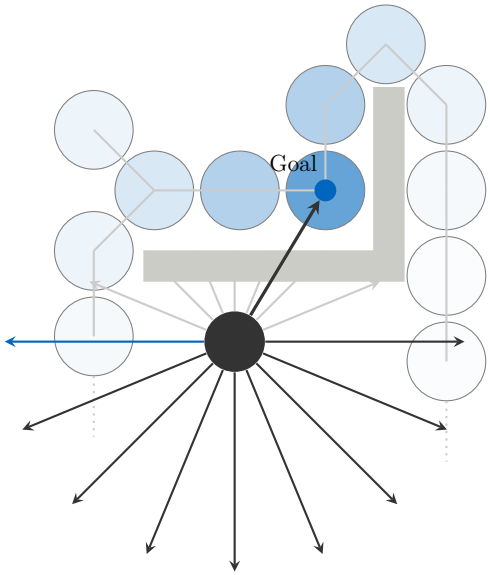


Figure 5.7.: The agent encountered an obstacle in the goal vector direction ($d < 0.6$m). It switches to topology-based navigation and performs a sub-goal localization. It starts a linear lookahead in all non-blocked traveling directions (dark grey thin vectors) and identifies the most promising one (blue vector).

When the agent encounters an obstacle in its heading or the goal vector direction ($d < 0.6$m), it switches to topology-based navigation. Relying on the information stored in the cognitive map, it tries to identify the optimal way around the obstacle.

We perform a directed linear lookahead in several desired traveling directions. Those can be the 16 directions in which we check for obstacles or a subset of them. It might make sense to only consider [North, East, South, West] as traveling directions for maze-like structures. If a direction θ is blocked - according to the definition depicted in Figure 5.5 - we disregard it. However, if θ does represent a valid traveling direction, we pursue the linear lookahead with velocity vector $v = |v| \cdot [cos(\theta), sin(\theta)]$, where $|v|$ is the lookahead speed.

Place cell and reward spiking values. For each time step of the lookahead, we determine the place cell firing p, according to the formula (5.2.2). The maximum reward spiking is then given by:

$$b = \max\{p \cdot r\} \tag{5.11}$$

where r is the reward cell vector. Like the linear lookahead along two axes, we save the reward spiking value b at the start of the simulation or if the value exceeds the previously saved value. The corresponding distance traveled is also given by $d = |v| \cdot dt \cdot k$ where k is the number of time steps dt passed. We then provide the velocity vector v as virtual input to the grid cell modules and start the next simulation step. Similar to the lookahead in two axes, we implemented an option to check the reward spiking only every n^{th} time step to speed up the algorithm.

Abort conditions. If we encountered a reward spiking $b_{max}^{0...k-1} > 0.8$ and the reward starts to become smaller again, it makes sense to stop the lookahead in that direction. We do so at the threshold of $b_k < 0.85 \cdot b_{max}^{0...k-1}$. The second abort condition is a timeout after enough time steps so that the lookahead could have traversed half of the environment.

Choosing the sub-goal. After each lookahead in a specific direction θ, we reset the grid cell modules to their original firing values. Once having simulated all directions, we determine the one with the highest reward spiking value b. Of all the locations encountered during the lookahead, the one closest to the goal is in that direction. This direction, therefore, corresponds to the most promising travel direction.

The agent pursues this direction until it reaches the sub-goal place cell or a new traveling direction becomes unblocked. In both cases, it performs a new sub-goal localization. If there is no obstacle nearby anymore (all $d_\theta > 1.3$m or the agent is close to the goal (the reward spiking of a lookahead exceeds 0.9), then we switch back to vector-based navigation.

6. Experiments

In this chapter, we want to evaluate how well the proposed algorithm performs. First, we will show that the implemented grid cell modules successfully form hexagonal patterns during initialization and can perform path integration. We will then test and compare three different approaches of decoding grid cell firings to receive a navigational vector. We also performed experiments to see how well a cognitive map can represent the environment. Based on that map, we will ask the agent to perform sub-goal localization and highlight several sample cases. Lastly, we will test the entire system in a maze proposed by Banino et al. 2018 and show that by combining vector-based with topology-based navigation, we can exploit shortcuts and navigate effectively to the goal.

6.1. Setup

The pybullet physics simulator is a python-based library capable of simulating a wide range of use-cases, including robotic movement and obstacle collision. It allows us to test our algorithm on a virtual robot and perform several experiments efficiently. We believe that the performance in the virtual environment and in real-world tests are strongly correlated. As a robot model, we choose the Pioneer P3DX, which implements differential drive and can detect obstacles ray-based in 16 equally spaced directions. We tuned the grid cells, place cells, and cognitive map to cover a circular environment with radius $r = 15\text{m}$. That environment size covers the maze dimensions of $11\text{m} \times 11\text{m}$ used in the final experiment. As we implemented the nested approach of combing grid cell modules as described in chapter 2 and section 5.2.1.2, we determined the minimum gain factor to be $g_m = 0.2$. We, therefore, ensure that at each location in the chosen environment, a unique set of grid cells fire. The time step size is $dt = 10\text{ms}$ and the agent travels with a constant speed of 0.5m/s. To ensure a stable time integration in the update rule of the grid cells, we had to limit the highest gain factor of the grid cell modules to $g_m = 2.4$. We decided to use six grid cell modules to allow for different grid cell orientations and scales to develop. According to the formula (5.2.1.2) this leads to gain factors of $g_m \in [0.2, 0.33, 0.54, 0.89, 1.46, 2.4]$.

6.2. Grid Cell Modules and Path Integration

We initialized the grid cells with random spiking values $s_i \in [0, 10^{-4}]$ and let the network evolve for 1000 time steps. At each time step we either applied no external input to the network with a probability of $p = 0.95$ or a small random velocity vector where v_x and $v_y < 0.2$m/s. After about 250 time steps, one out of the six modules already starts to show distinct spikes. By around time step 350, all modules have developed clearly defined spikes. Those spikes then start to rearrange themselves in a hexagonal pattern, and after a total of about 600 time steps, we could not recognize any noticeable changes in the grid cell pattern anymore. In Figure A.1 an example of such an initialization process is shown.

In each new initialization, a different pattern develops due to the random elements in the process. For the following experiments, we saved and reloaded the same grid cell state after an initialization of 1000 time steps. It is shown in Figure 6.1. We chose a state where the grid orientation differs between modules to ensure that the linear lookahead along the x and y-axis will work properly.

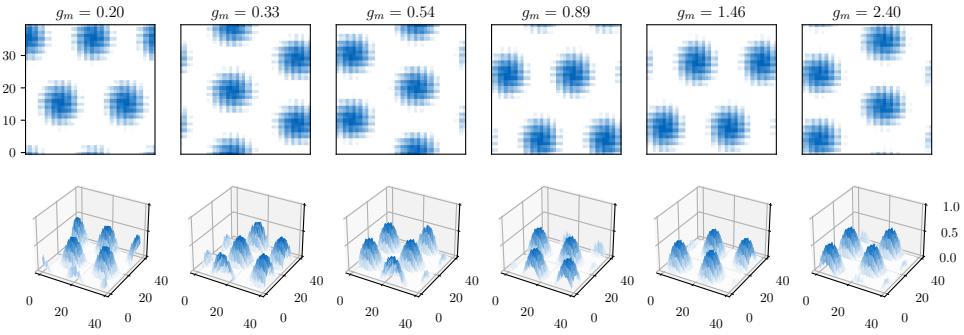


Figure 6.1.: Initialization of grid cell modules loaded before performing the following experiments. Note that the patterns are oriented differently across modules. In both the 2D projection and respective 3D visualization, blue indicates a high spiking value.

We then tested the path integration capability by applying a velocity vector of $\vec{v} = [0.5, 0.5]$ to the grid cell modules for 3000 time steps. The grid patterns successfully shift, as seen in Figure 6.2. We let the grid cell spiking stabilize during a break of 1000 time steps and then apply $\vec{v} = [-0.5, -0.5]$ for the equal time of 3000 time steps. The current spiking pattern starts to align with the initial pattern again. After the 3000 time steps and another pause, we compare both patterns. To do so, we make use of the previously described place cell implementation. We created a place cell at the beginning of the simulation, which initially has a spiking value of $p^{(0)} = 0.9383$. Note that this value is not

one as only grid cells with $s_i > 0.1$ form a grid-place cell connection. After the simulation that place cell spikes with $p^{(8000)} = 0.9185$. The overall similarity score then calculates as $p^{(8000)}/p^{(0)} = 97.89\%$, which is sufficient to perform long-range navigation.

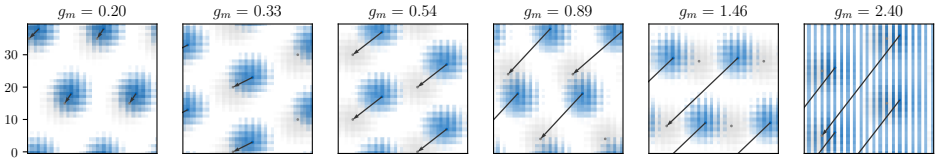


Figure 6.2.: Shift in grid cell pattern after applying $\vec{v} = [0.5, 0.5]$ for $k = 2000$ time steps to the grid cell modules. The grey colored pattern represents the initial state, the blue colored one the current state. The spikes are matched via vectors for visualization purposes. Note that the grid cell module with $g_m = 2.4$ seems to be unstable, yet recovers after a short pause.

6.3. Grid Cell Decoding

In the following experiment series, we want to test the precision of different grid cell decoding mechanisms. These include phase offset detectors, the linear lookahead in two dimensions, and spike detection. For each mechanism, we performed 50 trials. At the beginning of each trial, we choose a random point located 15m away from the origin. The agent is initially positioned at the origin and turns immediately in the direction of the chosen point. It then travels to that point with a constant speed of 0.5m/s. Once it comes close to the point, it slows down and comes to a halt. During the simulation, we apply the velocity vector as input to the grid cell modules. The time step size is $dt = 10$ms and by time step 3500 the agent has long reached its goal. We then switch to the navigation phase and compute the goal vector with one of the different decoding mechanisms. The agent heads in the direction of the goal vector and updates it along the way. After it covered a certain percentage of the return path (80% for linear lookahead or 5% for the other mechanisms), it will recalculate the goal vector fully. In the majority of cases, the agent finds its way back to the origin. The simulation is stopped after 8000 time steps.

We evaluate the performance of the three mechanisms according to several factors. Firstly, the overall distance Δ between the estimated goal location and the actual goal location (origin). Secondly, the angle deviation α between computed goal vector and actual goal vector and thirdly the deviation δ in their length. We evaluated these criteria after the initial vector computation at $d = 15$m as well as after the agent has recalculated the

vector at least once (at time step $k = 6300$ where $d \sim 2.5$m). We also check at the end of each trial if the agent was able to find the goal, meaning that its estimated goal location is $\Delta < 0.5$m away from the origin.

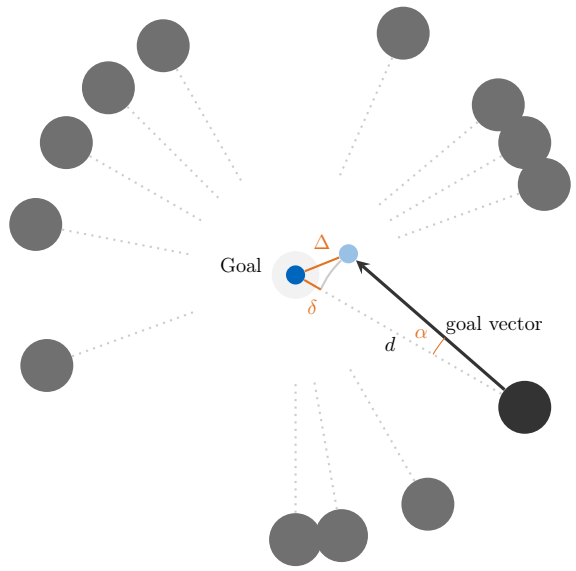


Figure 6.3.: Experiment setup to test the precision of several grid cell decoding approaches. The robot travels $d = 15$m away from the origin in a random direction. It then computes the goal vector back to the initial position. The absolute distance Δ between actual goal location and estimated location is considered as error. We also considered the deviation in angle α and in length δ between actual and computed goal vector. The experiment is performed 50 times for each decoding mechanism.

6.3.1. Linear lookahead in two dimensions

The linear lookahead is, as mentioned before, biologically plausible as it only relies on already found cell types and fulfills the evidence that the time needed for vector computation scales with the distance to the goal. However, this is also the biggest disadvantage of this mechanism. Each vector computation requires the agent to pre-play the grid cell shifts in 4 directions. The lookahead is aborted once the goal is found, yet for long distances, this still requires significant computational effort. Moreover, the maximum lookahead speed is limited, as the grid cell spiking otherwise becomes unstable. Potentially using more suited time integration methods could help to overcome that

issue and reduce synapses cost drastically. In our implementation, lookahead times were still significant, so that we had to limit the number of recalculations. The agent only recalculated the goal vector once it traveled 80% of the return path.

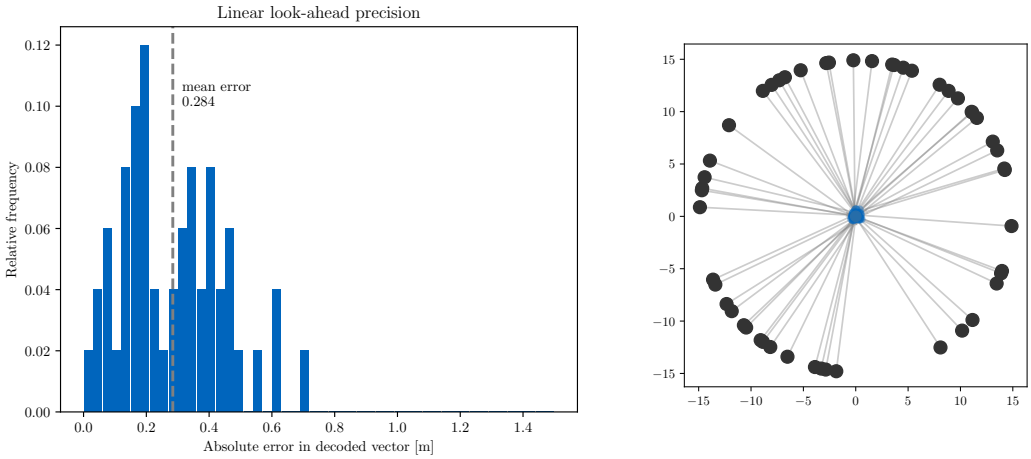


Figure 6.4.: Histogram of errors Δ of goal vectors computed with the linear lookahead mechanism from $d = 15$m. Mean average error computes as $\Delta_{avg} = 0.284$m. On the right all trials and their computed goal vector are shown.

The mechanism showed to be accurate independently of the distance. At $d = 15$m we measured a mean average error of $\Delta_{avg} = 0.284$m as shown in Figure 6.4. At $d \sim 2.5$m we identified $\Delta_{avg} = 0.278$m and at the end of the simulation of $\Delta_{avg} = 0.259$m. Additionally, 92.0% of all trials ended with $\Delta < 0.5$m, and the furthest $\Delta = 0.607$m away. Both angle and distance computation of the goal vector seem to work equally well. Understandably, in contrast to the distance estimation, the angle estimation becomes more imprecise the closer the agent gets to the goal. Overall the linear lookahead in two dimensions proved reliable and sufficiently accurate, even across long distances.

6.3.2. Phase Offset Detectors

We followed the instructions of Edvardsen 2015 closely to rebuilt the proposed phase-offset detector network. Each detector is tuned to respond to a shift between goal and current grid cell spiking in a specific direction. The detectors are spread out in a regular grid structure across the neuron plane. We decided to use a grid of $S_{xy}^2 = 9 \times 9$ locations, each holding $S_\theta = 16$ detectors for the different directions. We calculated the activity of

the motor-output neurons (each corresponding to one of the 16 directions) strength as proposed by Edvardsen 2015 and described in chapter 3.1. To compute the goal vector Edvardsen 2015 suggests to sum over all directional vectors $\hat{e}_{\theta_k}$ scaled by the motor neuron activity u_k and then adjust the overall vector length by a factor of ρ:

$$\Theta = \rho \cdot \sum_k u_k \cdot \hat{e}_{\theta_k}, \quad \rho = \frac{1}{S_\theta \cdot S_{xy}^2 \cdot \sum_m g_m^{-1}} \tag{6.1}$$

For our chosen parameters ρ calculates as $\rho = 6.37 \times 10^{-5}$. However, after initial tries, we realized that the computed goal vector lengths do not match the actual goal distance. We decided to use $\rho = 0.27$ for all of our trials by performing an initial parameter search. We validated our initial choice by computing the mean squared error of all deviations δ between the actual goal distance and the computed vector length. The mean squared error across all trials is minimized with $\rho = 0.2853$, as it can be seen in A.2. This result validates our choice of $\rho = 0.27$, and we can therefore consider the following performance analysis as meaningful.

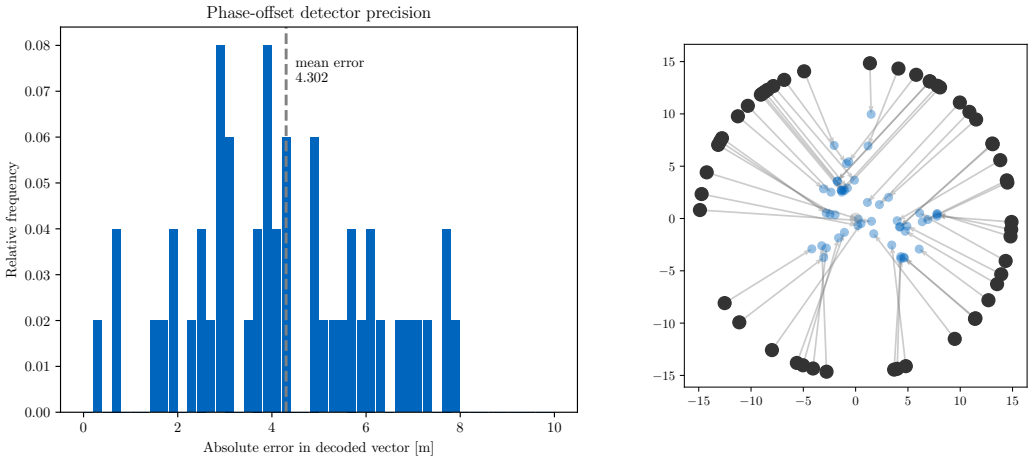


Figure 6.5.: Histogram of errors Δ of goal vector computed with the phase-offset detector mechanism from $d = 15$m. Mean average error computes as $\Delta_{avg} = 4.302$m. On the right all trials and their computed goal vector are shown.

For long distances ($d = 15$m) the phase-offset detectors showed a considerable error of $\Delta_{avg} = 4.302$, as portrayed in Figure 6.5. However, the detected direction was roughly correct ($\alpha_{avg} = 6.589°$, which allowed the agent to approach the goal successfully. The error in the computed goal vector got considerably smaller with decreasing distance

($\Delta_{avg} = 0.845$) and given the small computational effort, we were able to recalculate the goal vector very frequently. 94% of all 50 trials ended with a $\Delta < 0.5$. This matches with the results of a similar experiment performed in Edvardsen 2015. An agent had to return to the origin from a random location up to about 10m away. Out of 500 trials, 6 did not end up in $\Delta < 0.5$, corresponding to an overall return rate of 98.8%.

6.3.3. Spike Detection

The spike detection mechanism is an image processing approach and, therefore, not biologically plausible. However, it can serve as an additional reference to comparing the performance of the different decoding mechanisms. We processed the grid cell neuron sheets as images of 40×40 pixels per module and identified the peaks of both current and goal spiking patterns. We then matched the peaks of both patterns by starting with the module with the smallest g_m and simply matching the peaks closest to one another. We average over the vectors connecting the current with the goal peak and retrieve our first unscaled estimate of the goal vector, denoted by $\hat{\Theta}_m$. To analyze the next module with index i, we then scale that vector by a factor of $g_m^{(i)}/g_m^{(i-1)}$ and look for matches within that direction and distance. By iterating through all modules up to the ones with a higher resolution, we can further increase the precision. Finally, we compute the overall goal vector Θ as a weighted average of the module-specific unscaled goal vectors $\hat{\Theta}_m$. The modules with higher g_m have an accordingly higher weight:

$$\Theta = \rho \cdot \frac{1}{\sum g_m} \cdot \sum_m \hat{\Theta}_m \tag{6.2}$$

where we used $\rho = 1$ for the experiments. We did the same mean error analysis as for the phase-offset detectors and determined that $\rho = 0.7475$ would minimize the mean error. However, as shown in Figure A.3 a value of $\rho = 1$ is still acceptable and is the value considered for the following results. It is good to though to keep in mind that adjusting ρ in future experiments could increase the performance.

The spike-detector mechanism delivers acceptable yet not very reliable results. Over the distance of $d = 15$m we recorded an error of $\Delta_{avg} = 4.877$, as portrayed in Figure 6.6. The agent does not head to the goal in all trials, leading to a split error distribution for smaller d. If we do not consider the outliers (10% of all trials, refer to Figure A.4), the results are comparable to the phase-offset detectors. However, overall only 60% of all trials achieve a $\Delta < 0.5$ at the end of the simulation. The performance could be improved by increasing the reliability of the spike matching algorithm and adapting ρ as discussed.

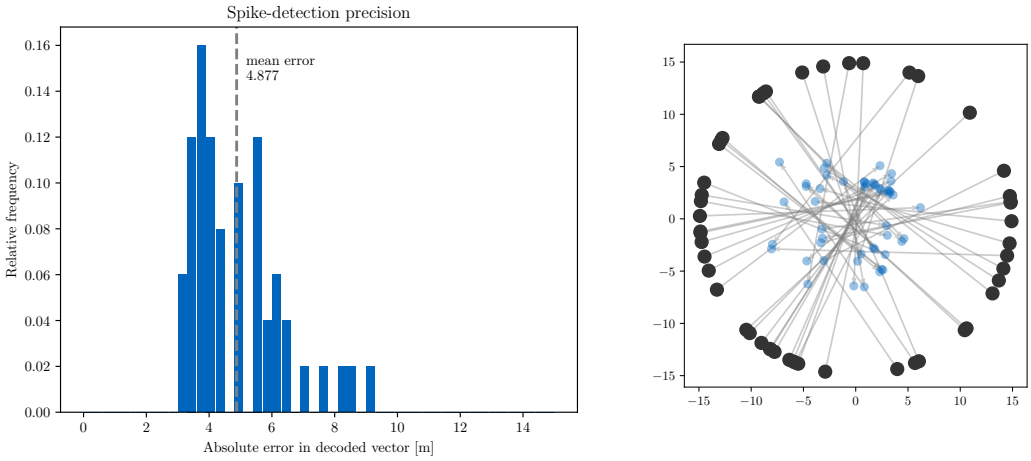


Figure 6.6.: Histogram of errors Δ of goal vectors computed with the spike-detector mechanism from $d = 15$m. Mean average error computes as $\Delta_{avg} = 4.877$m. On the right all trials and their computed goal vector are shown.

6.3.4. Comparison

In the table 6.1 we compare the experiment data of the three decoder mechanisms. Overall, the linear lookahead decoder performs best. It is biologically plausible, the computed goal vectors are precise even across long distances, and almost all trials end in proximity to the goal. Its disadvantage is, however, the computational time that scales with the goal distance. A biologically plausible alternative is the phase-offset detector mechanism. It allows for frequent vector recalculations and delivers acceptable results, especially for smaller distances. The spike-detector mechanism passed as proof of concept. It can be further improved to reduce the number of outliers. If we do not consider these outliers, it delivers similar results as the phase-offset detector mechanism. Given the results, we decided to use the linear lookahead decoder mechanism for the remaining experiments.

Decoder			linear look-ahead	phase-offset detector	spike-detector
$d =$ 15m	Δ_{avg}	[m]	0.284	4.302	4.877
	δ_{avg}	[m]	0.224	3.830	4.463
	α_{avg}	[°]	0.511	6.589	5.485
$d \sim$ 2.5m	Δ_{avg}	[m]	0.278	0.845	1.759 | 0.786*
	δ_{avg}	[m]	0.178	0.755	1.719 | 0.747*
	α_{avg}	[°]	4.315	5.467	9.065 | 3.493*
end	Δ_{avg}	[m]	0.259	0.397	1.276 | 0.423*
	$\Delta < 0.5$	[%]	92.0	94.0	60.0

Table 6.1.: Comparison of grid cell decoder mechanisms. The (*) indicates the result where outliers ($\Delta > 2.5$m, $10 - 15\%$ of all trials) were not considered. For the definition of $\Delta, \delta, \alpha, d$ refer to Figure 6.3.

6.4. Cognitive Map

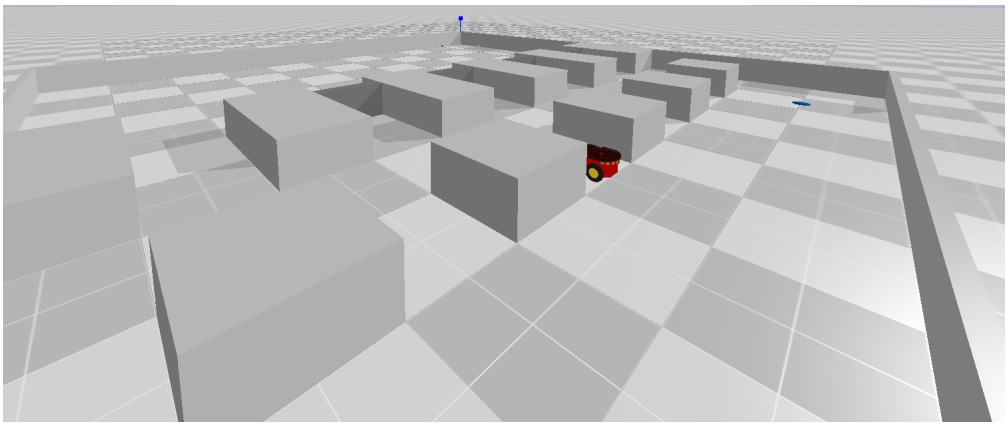


Figure 6.7.: Maze and agent in pybullet environment. The maze is sized 11×11m and its layout is inspired by Banino et al. 2018.

Maze configuration. A vector-based navigational approach is most efficient in an environment without obstacles, as the agent can head straight to the goal. The advantages of a topology-based or combined approach only come into play once we add obstructions to the map. For our experiment, we chose the linear sunburst maze inspired by Banino et al. 2018. In the lower section of the maze, the agent can move freely, while in the upper section, equally spaced blocks limit the paths it can travel. Five doors connect the two sections and can be open or closed between the exploration and navigation phase. We

implemented the maze for the pybullet simulator and scaled it to the size of 11×11m. These dimensions give the agent enough room to travel between the obstacles. A graphical visualization of the environment simulation can be seen in Figure 6.7. We chose the lower-left corner of the rectangle as the origin and can therefore specify the agent's initial position as $(5.5, 0.5)$ and the goal location as $(1.5, 10)$.

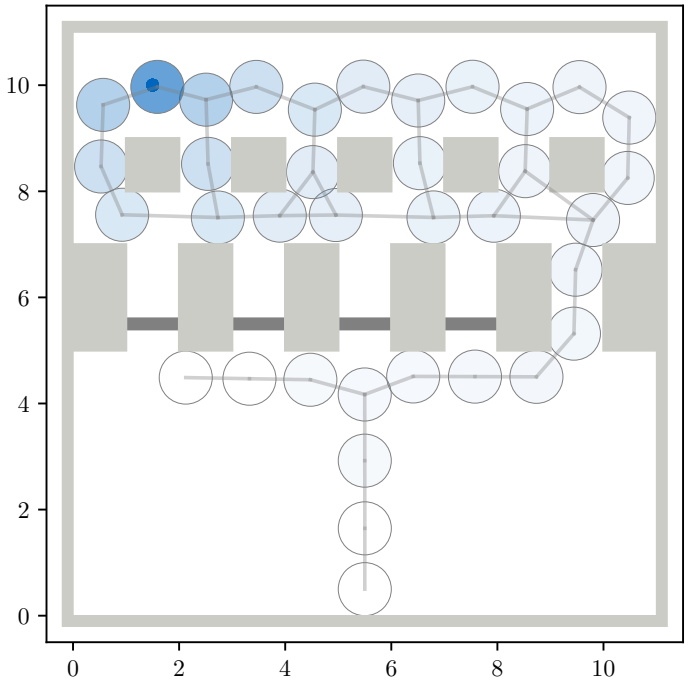


Figure 6.8.: Cognitive map after hard coded exploration phase of 15000 time steps. Obstacles are colored light gray and doors one to four (medium gray) are still closed. The goal is marked by the blue dot. The circles represent place cells, the connections between them visualize the topology network. The coloring of the place cell is correlated to the reward value of the corresponding reward cell. The darker the blue, the higher is the reward value.

Exploration Phase. For the exploration phase, we closed all doors except the last one (number five). We decided to precode a trajectory for the agent to ensure that it will travel through all relevant areas of the maze and encounters the goal. The agent travels straight up to the doors, then heads West along the closed doors, and then back to the open door. It traverses to the upper section and covers all sub-paths there. As the agent traverses the environment, it successfully creates place cells as expected. With a place

cell firing threshold of $p_{max} < 0.85$, the distance between the center of two adjacent place-fields is around 1.2m, which is a suitable resolution for our purposes. The recency cells kept track of the last occasion of visit. The parameters described in section 5.2.3.1 ensured that in the topology cell network, we connect place cells that are close to each other even if in between the agent might have to make a turn. The parameters are also such that we do not connect place cells that are too far apart from each other. Once the agent finds the goal, we let it propagate through the reward cells up to a depth of 15 levels. Figure 6.8 shows the cognitive map at the end of the exploration phase.

Cognitive Map precision. Whenever we created a new place cell, we linked it to the current position of the agent, allowing us to create a visual representation of the place cell and cognitive map network. However, we were unsure if a drift between the position saved and the position encoded by the place cell can occur. To determine the encoded position, we performed a linear lookahead from the initial position to each place cell and compared the computed vector to the saved position vector. The computed vectors and the error distribution are shown in Figure 6.9. We observed a mean average error across all 36 place cell positions of 0.383m. Given that the linear lookahead mechanism has a mean average error of about 0.28m in itself, we conclude that only a slight drift in the place cell position occurs and that our cognitive map is suitable to be used for navigation.

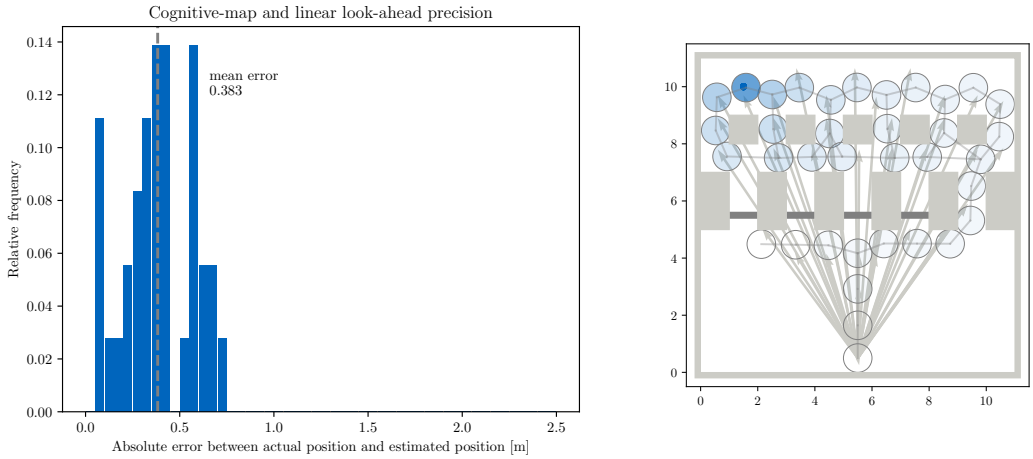
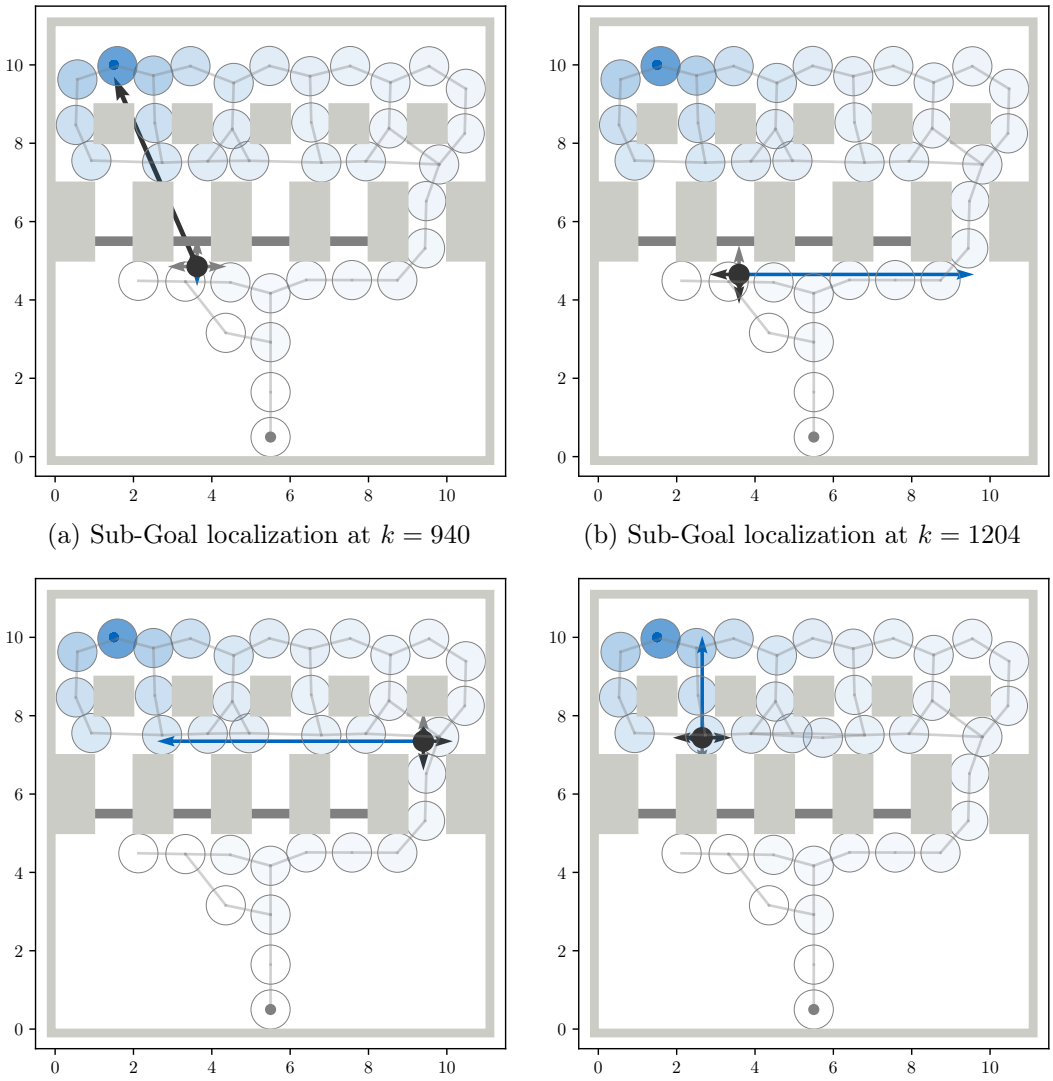


Figure 6.9.: We used the liner lookahead to decode the position vector corresponding to a place cell (right picture). We compared them to the actual position of place cell creation (left plot) and calculated a mean average error of 0.383m.

6.5. Sub-Goal localization



(a) Sub-Goal localization at $k = 940$

(b) Sub-Goal localization at $k = 1204$

(c) Sub-Goal localization at $k = 3024$

(d) Sub-Goal localization at $k = 4399$

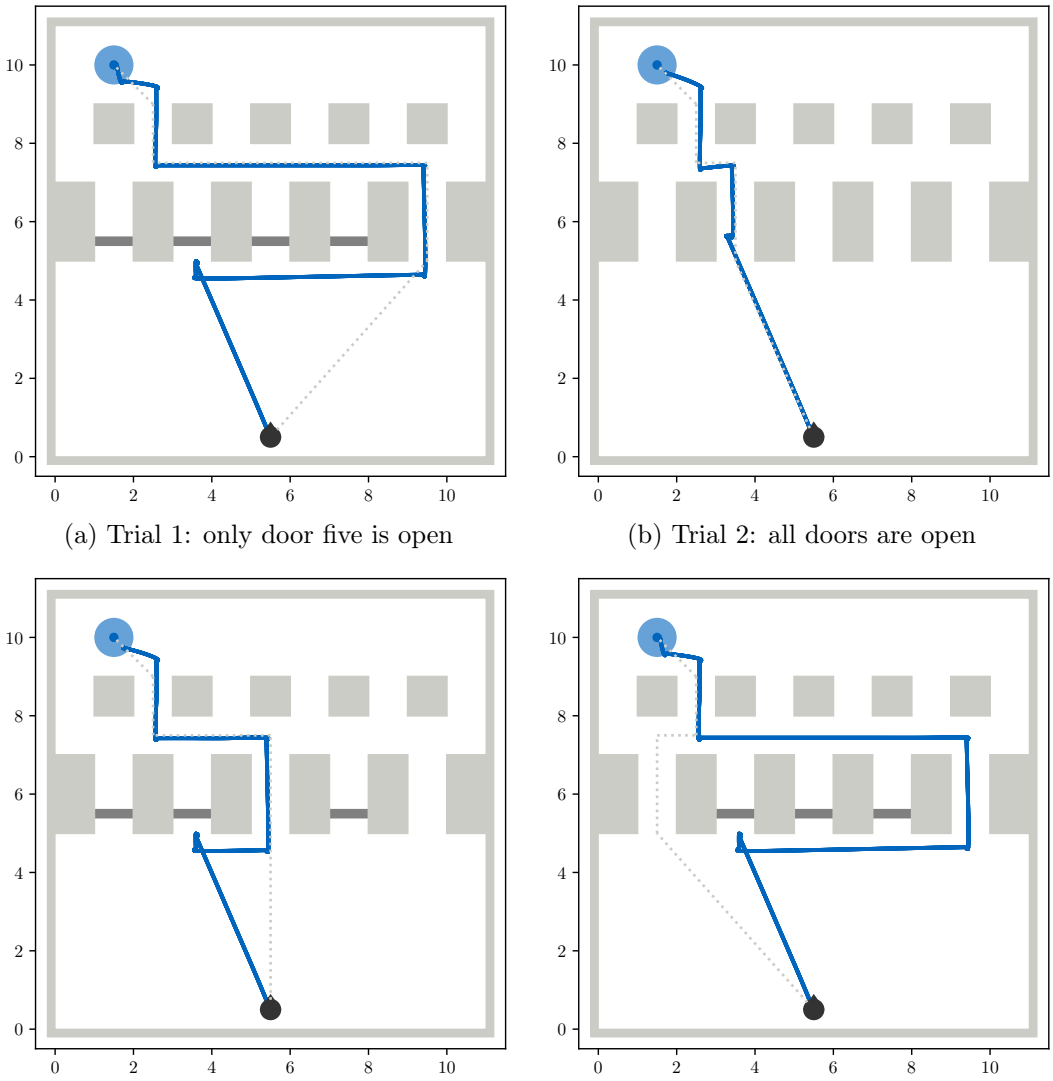
Figure 6.10.: Examples of sub-goal localizations through performing a linear lookahead in four directions. Direction with highest reward spiking (blue), blocked directions (grey), other non-blocked directions (dark grey). In Figure 6.10a, the dark grey vector corresponds to the actual goal vector. Lookahead data can be found in A.1.

The agent starts the navigation phase by computing the goal vector and heads in that direction. Once the agent encounters an obstacle, it switches to topology-based navigation. Based on the obstacle detection mechanism described in section 5.3, we determined what travel directions were blocked and which were free. Given that the maze consists of densely clustered, right-angled obstacles only, we decided the only meaningful travel direction during topology-based navigation would be North, East, South, and West. Note that in other environments, it might make sense to consider other directions as well. We performed a directed linear lookahead as described in 5.3.2 in all none blocked directions and picked the one with the highest reward spiking. It was sufficient and less computational intensive to recalculate the reward spiking only every 40^{th} time step, which corresponds to a virtually traveled distance of $d = 0.2$m. Based on the cognitive map, the agent picked the logically right choice in all sub-goal localizations we performed across the different trials. Figure 6.10 shows four examples of sub-goal localizations, and their corresponding look ahead-data can be found in appendix A.1.

6.6. Overall performance

We tested our proposed algorithm in five different configurations of the maze, depicted in Figure 6.11 and 6.12. In each, different doors are open, and collectively they give us hints on the strengths and improvement potential of the system. In the following, we will elaborate on the individual trials and finally compare them in the overview table 6.2.

Navigation in a known environment. For the first trial, we kept the configuration the same as during the exploration phase, so only door five was open. The agent initially performs vector navigation. To reduce computational effort we decided to check the reward spiking during the linear lookahead in two axes only every 10^{th} time step. This represents a resolution of 0.05m, which given the size of the maze is highly sufficient. The agent finds the goal vector and heads in the direction of the goal. It encounters closed door number two and switches to topology-based navigation. It identifies the direction East as most promising, as it leads to a location it has been before. Whenever the agent reached the sub-goal place field or a new travel direction (North, East, South, West) becomes available, it performs a new lookahead to determine the next sub-goal. Given that the environment has not changed compared to the exploration phase, the agent follows the reward gradient in the cognitive map. Close to the goal, the reward spiking exceeds the threshold of > 0.9, and we assume that no obstacle is in the agent's way. It switches back to vector-based navigation and approximates the goal directly. The agent reaches the goal with an overall distance traveled of 26.3m. However, if we do not consider the back and forth at the very end to get as close as possible to the goal, we



(a) Trial 1: only door five is open

(b) Trial 2: all doors are open

(c) Trial 3: doors three and five are open

(d) Trial 4: doors one and five are open

Figure 6.11.: We tested the our proposed algorithm in different maze configurations. The dark grey rectangles are closed doors. The agents trajectory is depicted in blue, the shortest available path is grey dotted. Another trial is shown in Figure 6.12.

measured a distance of 24.5m until the first goal encounter. The shortest available path is 18.4m long, given that the agent would have head to door five directly. So overall,

the path the agent picked was 33.2% longer than the optimum. A purely vector-driven agent such as the one proposed by Banino et al. 2018 would head to door one as well, before eventually, by trial and error, finding its way to door five. This search increases the path length significantly, proving that a topology-based navigational component is highly valuable. In the following, we will see examples of when the vector-based approach can have advantages.

Exploiting shortcuts. For our second trial, we opened all doors. The agent again heads to door two, and once it encounters the side of the tunnel, it looks for a sub-goal. It finds the newly available shortcut and heads North. Once back in know territory, it proceeds as in trial one. With this approach, the agent found a path only 3.5% longer than the shortest one. Compared to the shortest path from the original configuration, the agent only traveled 63.6% of the distance. The system proposed by Banino et al. 2018 shows similar behavior in this maze configuration and is also able to exploit shortcuts. However, in about 60% of the runs, it chooses door one, which is (as an idealized path) 9.7% longer than the alternative through door two.
For our third trial, we opened only door five and door three. The agent heads to door two first but then chooses to take the shortcut through door three once it passes by. The path length is therefore only 87.5% of the shortest path of the original configuration. These results prove that a combination of vector and topology-based navigation can be highly effective.

Missing a shortcut. In our fourth trial, we opened doors one and five. The path through door one is the shortest in this configuration, yet the agent chooses to travel to door five. When switching to topology-based navigation in front of door two, it faces the decision to travel East, which is a known path, or West, which proved unsuccessful during the exploration phase. The agent still reaches the goal, but its path is 97.6% longer than the shortest available path. The vector-based agent described in Banino et al. 2018 is instead driven to door one and rarely misses that shortcut if available.

Blocking a known path. For our fifth and final trial shown in Figure 6.12, we only opened door one and closed all others, including door five. As in the previous trial, the agent misses trying door one and heads to door five. Here the agent gets stuck as the door is closed. Even if the agent randomly distances itself from the door, once a new lookahead is started, the reward gradient in the cognitive map leads the agent back to door five. The agent, therefore, did not reach the goal by the end of the simulation.
To enable the agent to deal with blocked paths, we would have to delete the corresponding topology cell connection and recalculate the reward diffusion. The agent would then eventually find the way through door one. In future work, the system could be extended to be capable of finding and weakening connections (in a biologically plausible manner) if such situations occur.

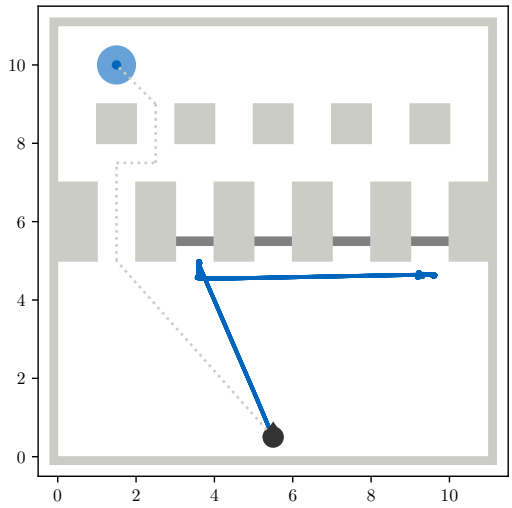


Figure 6.12.: Trial 5: only door one is open

We summarized the results of all trials in Table 6.2. The agent always reached the goal, except when prior available paths were blocked. Overall, combining vector and topology-based navigation proved to be an effective approach across various maze configurations.

Doors open		only 5	all	5 and 3	5 and 1	only 1
Goal reached	-	yes	yes	yes	yes	no
Distance overall	[m]	26.3	11.7	16.1	26.3	12.3
Distance to goal	[m]	24.5	11.7	16.1	24.5	-
Shortest path	[m]	18.4	11.3	12.9	12.4	12.4
× shortest path	-	1.332	1.035	1.248	1.976	-
× known path	-	1.332	0.636	0.875	1.332	-

Table 6.2.: Overview of performance in different maze configurations. The multiples are based on the distance traveled until the goal was first encountered. With known path, we refer to the shortest path in the maze configuration of the exploration phase (only door five open).

7. Discussion

In this chapter, we will critically evaluate the outcome of our experiments. First, we will discuss if and how the proposed system can be used for robotic applications and where we see its strengths and weaknesses. Secondly, we will assess our work regarding biologically plausibility and highlight where we had to make assumptions.

7.1. Robotic applicability

Capabilities. The results from our maze experiment prove that the proposed system achieves its objective. The agent can create a map of the environment and remember the location of the goal. It reliably returns to the goal and exploits shortcuts on the way. However, in the configuration where we blocked a previously available path, the agent fails to reach the goal. This problem can be solved by removing the respective topology cell connections within the cognitive map.

Technical requirements. The proposed algorithm requires only little external input. The primary input is the current speed and heading direction of the robot. Additionally, a lidar sensor detecting obstacles in 16 equally spaced directions is required. However, an alternative obstacle detection mechanism based on other sensory information would also be feasible. Furthermore, to store the cognitive map, no extensive memory capabilities are needed. Nevertheless, the computational efforts during navigation are considerable. Simulating the grid cell firing based on the NumPy library is relatively fast. Still, during a linear lookahead for goal or sub-goal localization, we have to preplay several trajectories with up to 3000 computational steps. We can reduce the number of steps on the cost of precision, yet computational effort will remain to scale with the goal-vector length. If the algorithm has to run at a very low cost, it might make sense to consider the phase-offset detector mechanism from Edvardsen et al. (2020) instead. Overall, it remains questionable if our biological plausible approach outperforms a plane path integration or SLAM algorithm. Replicating neuron activity adds additional computational weight, which is hard to compromise for on other ends.

Environment properties. So far, we have only tested our approach in a maze-like environment with rectangular obstacles and straight paths. In this environment, it proved

effective to only consider North, East, South, and West as traveling directions during topology-based navigation. Doing so reduces the number of lookaheads per sub-goal localization and the overall number of sub-goal localization. This is because the agent initiates a new sub-goal localization whenever a new traveling direction becomes available. More complicated obstacle arrangements might require the agent to consider a broader set of traveling directions or lead to an unfavorable state where the agent gets caught up in sub-goal localization loops. However, this is still something to be investigated. We also want to note again that we have performed our experiments in a virtual environment only. A transition to an actual robot might lead to unexpected noise or similar. Still, we are confident that the overall performance will be comparable to the results achieved so far.

7.2. Biological plausibility

Our work aims for biological plausibility, and we suggest that the path integrator, the cognitive map, the vector decoding mechanism, and sub-goal localization achieve that objective. As path integrator, we used a continuous attractor network model simulating grid cell firing concurrent to biological findings (Burak and Fiete 2009). Moreover, the cognitive map is solely based on neurological circuitry. The grid cell firing propagates through the remaining network of place cells and prefrontal cortex cells. We only used basic NumPy operations to simulate these neurological mechanisms. Furthermore, with the linear-lookahead, we decided on a highly plausible grid cell decoding mechanism. It makes use of the existing cell architecture, and as recorded in rodents (Spiers et al. 2018), vector computation time scales with the vector length. Also, that the agent stops at obstacles and locates a sub-goal through replay has been suggested in earlier work. "However, there seems to be a diversity of different forward/reverse replay/preplay phenomena"(Edvardsen et al. 2020). In our work, we decided to simulate forward trajectories, as proposed by Erdem and Hasselmo (2012). Further investigations are needed to determine which mechanism is used by rodents.

We also want to highlight that our model of prefrontal cortex cells is supported by the neurological recordings described by Place et al. (2016), Burton et al. (2009) and Hok et al. (2005). Yet, we assumed that specific cell types such as reward cells, topology cells, and reward cells exist. Further research on the prefrontal cortex is necessary to potentially find these or similar cell types in the context of memory creation and retrieval. Moreover, we are aware that our model is not fully comprehensive yet. The velocity vector, for example, could be represented by speed and head direction cells. We also do not exploit motor cells or border cells. Refer to Section 8.2 for suggestions on how our proposed system could be extended.

8. Conclusion and Future Work

In this chapter, we will summarize our work. We will also give an outlook on the future work that could be done to develop our ideas further.

8.1. Conclusion

In this research, we aimed to develop a biologically inspired navigational system combining vector-based and topology-based navigation. The system had to be usable on mobile robotic platforms. It should enable a robot to explore its environment and navigate to a goal location it had previously found. With our work, we have successfully proposed such as system.

As primary input, we considered the current velocity vector of the agent. In analogy to biological findings, we use a network of grid cells to perform path integration based on that velocity vector and therefore keep track of the agent's current position. These grid cells are organized into modules, which differ in their grid scale. We combined six grid cell modules to encode the agent's location uniquely in large-scale environments. The largest grid scale determines the maximum environment size, while the smaller grid scales enhance the precision.

The grid cells connect to place cells, which fire only at a specific location in space. The agent creates a new place cell whenever it visits an area in the environment it has not been before. To each place cell, we link a recency cell, a topology cell, and a reward cell, which all represent prefrontal cortex cells. Recency cells encode when the agent has visited the place field last. Topology cells keep track of which place fields are located in proximity. Reward cells encode how close the place field is to a previously found goal location. All of these cells together allow the agent to represent the environment as a cognitive map.

For navigational purposes, we embedded the described cell network in a more extensive architecture. The robot, by default, starts with vector-based navigation and identifies a goal vector by comparing the grid cell spiking at the current and the goal location. We tested three grid cell decoders and showed that the linear lookahead decoder is most

precise, especially across long distances. When the agent encounters a goal, it switches to topology-based navigation. It identifies a suitable sub-goal to avoid the obstacle by performing a linear-lookahead in unblocked traveling directions. It chooses the traveling direction that will lead to the highest reward spiking within the cognitive map and, therefore, to a place field relatively close to the goal. The agent will continue with topology-based navigation until it is clear of obstacles around.

Our experiments in a virtual maze environment prove that an agent equipped with our proposed system reliably reaches the goal location. It also showed that combining vector-based and topology-based navigation allows the agent to circumvent obstacles while exploiting shortcuts when they become available. Our system only failed when previously available paths became blocked. On the technical side, it does not require any extensive memory usage and only little sensory input. However, the linear lookahead decoder is computationally expensive, given that its computational cost scales with goal-vector length. Moreover, it remains questionable if, for robotic application, simulating biological neuron behavior is worth the additional computational effort or if similar results can be achieved with other, more traditional algorithms.

Nevertheless, our biologically plausible approach can help to further understand the fundamental mechanism in the rodent's brain during navigation. We suggest that the path integrator, the cognitive map, the vector decoding mechanism, and sub-goal localization are inspired by neurological findings. We believe that the underlying principles of our system can guide further research in this area which will lead to more and more accurate models replicating biological behavior.

8.2. Future Work

We want to highlight several ways to develop our ideas further.

Additional cell types. In our model, we did not include all the cell types described in Section 2.3. Border cells could be helpful for obstacle detection. Moreover, involving them in the cognitve map could allow the agent to remember the position of an obstacle. With this information the agent could abort a lookahead earlier if it leads to an obstruction. Border cells could also help to recalibrate grid cell and place cell firing. Furthermore, head direction cells, paired with speed cells, could encode the velocity vector. Doing so would increase the biological plausibility, given that the interplay of head direction cells and grid cells has been studied before. Yet, it is questionable if this makes sense for robotic applications, given that either way, we somehow have to determine the agent's velocity through sensory input. Additionally, motor cells could be used to implement a motor controller based on neurological circuitry only.

Other improvements. We already mentioned that deleting a connection in the cognitive map would enable the agent to deal with blocked paths that had been available previously. Furthermore, we propose adding a navigational mode for when the agent encounters an obstacle at an acute angle. Edvardsen et al. (2020) implemented such a mode where the agent avoids the obstacle through a slight course adjustment and does therefore not need to switch to topology-based navigation. Another idea to improve performance is to save lookahead information between two consecutive sub-goal localizations. The agent does not necessarily have to preplay the trajectory it has been following or perform a lookahead in the direction from which it came. Moreover, we believe that adding a vision module would highly improve navigational capabilities. So far our agent practically found its way in the dark. Visual cues could help to recalibrate grid cells and to detect obstacles earlier on.

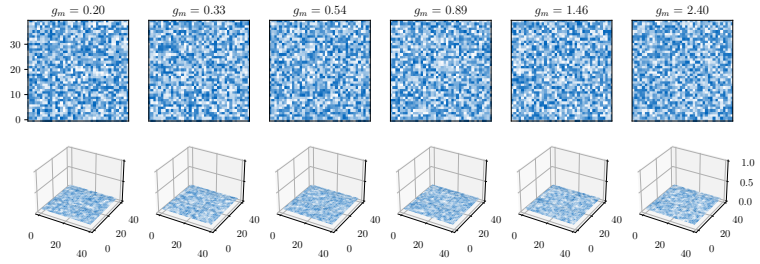
Further research. Our ideas would highly profit from further investigations in the functionality of the prefrontal cortex. As highlighted, it plays a central role in memory creation, which is crucial for creating a cognitive map. We want to raise the following questions: What navigational information exactly is stored in the prefrontal cortex? How is it updated and how can connections be added or deleted? How long is the information stored? Which cell types are involved?

With our work, we have proposed a biologically inspired navigational system combining vector-based and topology-based navigation. This system can now be extended and improved further as new biological findings arise.

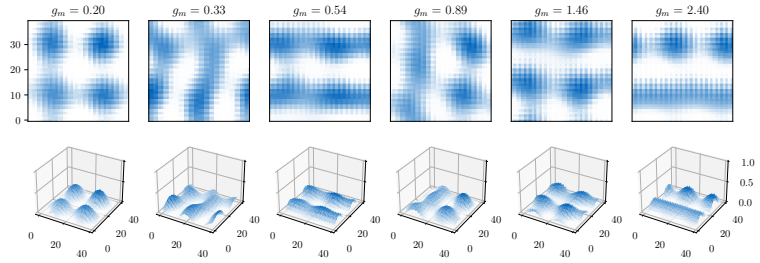
A. Supplementary Figures

Sub-goal localization		angle	distance [m]	step	reward	pc
$k = 940$	south	$3\pi/2$	0.0	0	0.000	0
$k = 1204$	east	0	6.0	1200	0.080	10
	west	π	0.0	0	0.000	0
	south	$3\pi/2$	0.0	0	0.000	0
$k = 3024$	east	0	0.4	80	0.103	12
	west	π	6.8	1360	0.284	27
	south	$3\pi/2$	0.0	0	0.099	12
$k = 4399$	east	0	0.2	40	0.285	27
	north	$\pi/2$	2.6	520	0.863	29
	west	π	0.2	40	0.285	27

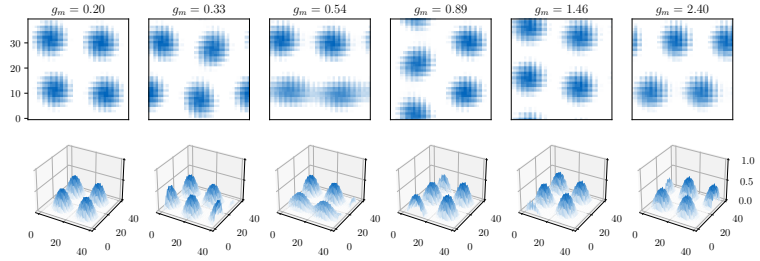
Table A.1.: Sub-goal localization data for examples shown in Figure 6.10. k is the time-step at which the localization takes place. Only data for non-blocked directions is listed. We determine the highest reward along each direction, list the place cell (pc) firing and at which time step of $dt = 10$ms it occurred. Note that we only reevaluate the reward spiking every 40^{th} step. The look-ahead speed is 0.5m/s and with it we can calculate the distance. Finally, the direction with the highest reward spiking value is chosen.



(a) $k = 0$: Grid cell modules are randomly pre-initialized with low spiking values.



(b) $k = 250$: First spikes develop in the modules and will from now on separate further.



(c) $k = 350$: All the modules have established clearly defined peaks. In the remaining phase they will rearrange in a hexagonal pattern.

Figure A.1.: Initialization of grid cell modules. The development of their state is shown at several different points in time. In both the 2D projection and respective 3D visualization, blue indicates a high spiking value.

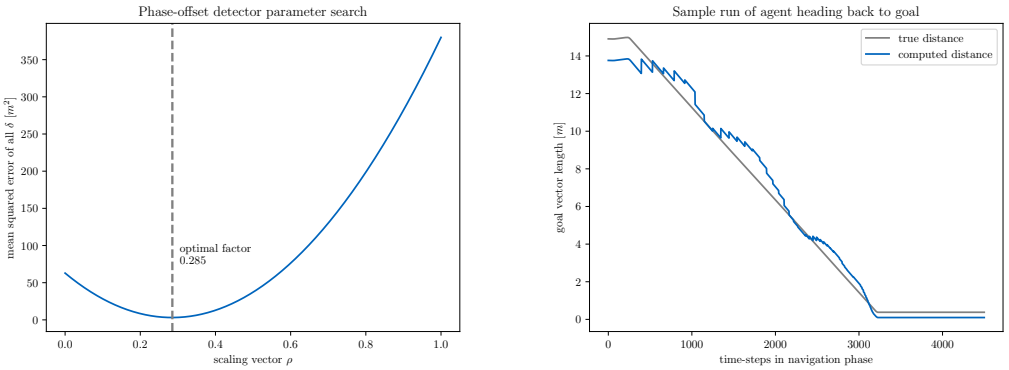


Figure A.2.: The goal vector computed by the phase-offset detector network has to be scaled with a factor of ρ. We calculated the mean squared error across all 50 trials between the actual distance to the goal and computed vector length. $\rho = 0.285$ minimizes that error. On the right, a single trial is shown, where the agent navigates back to the origin. Distances are adapted to match $\rho = 0.285$. While at the beginning the distance is slightly underestimated, it is overestimated at later times. With our analysis we tried to minimize the errors in both directions, across all trials.

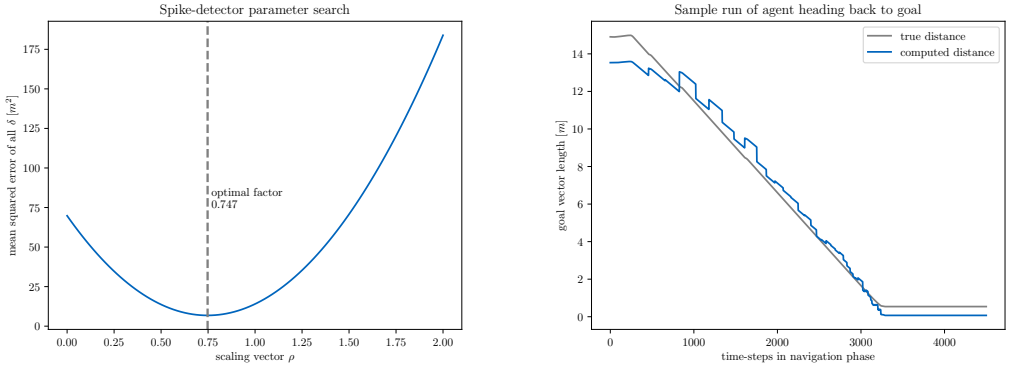


Figure A.3.: The goal vector computed by the spike-detector mechanism has to be scaled with a factor of ρ. We calculated the mean squared error across all 50 trials between the actual distance to the goal and computed vector length. $\rho = 0.747$ minimizes that error. On the right, a single trial is shown, where the agent navigates back to the origin. Distances are adapted to match $\rho = 0.747$. While at the beginning the distance is slightly underestimated, it is overestimated at later times. With our analysis we tried to minimize the errors in both directions, across all trials.

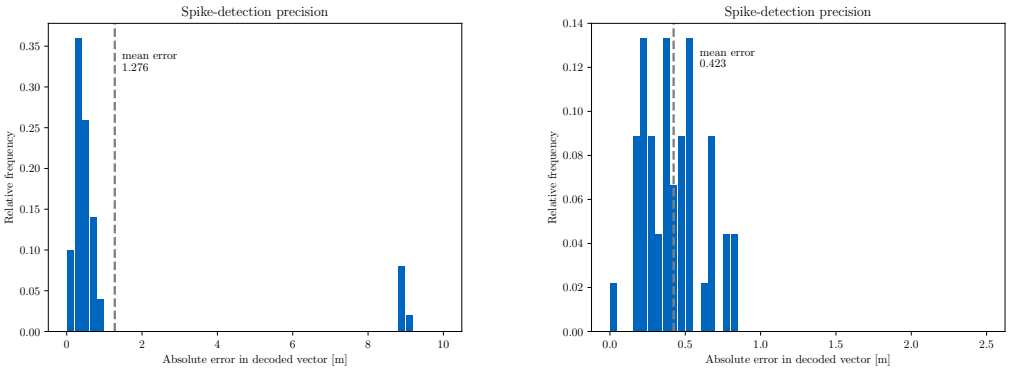


Figure A.4.: Histogram of distance Δ between computed and actual goal location at the end of each trial using the spike-detector mechanism. For all trials (left) the mean average error computes as $\Delta_{avg} = 1.276$m. However, if the outliers ($\Delta > 2.5$m $\equiv 10\%$ of all trials) are neglected (right), the filtered mean average error becomes $\Delta^*_{avg} = 0.423$m.

List of Figures

2.1. As the rat traverses a linear path, a grid cell will spike every time after traveling a distance of λ. By combining several grid cells (GC) with the same grid-scale λ but a phase offset of ϕ, the space is discretized. 4
2.2. Firing of a single grid cell from the entorhinal cortex of the rat brain. Left, the traveling trajectory of the rat is shown in black, spike locations in red. Right, the cell activity is shown as heat map, where red represents high activity and blue low activity. Figure adapted from E. I. Moser, Kropff, et al. (2008). 4
2.3. Discretization of the one dimensional space by a combination of two grid cell modules, m_1 and m_2. The colors indicate which of the six grid cells within each module is most active at the respective location. In this example, both approaches discretize the axis for a distance of d unambiguously. (a) m_1 has a relatively large grid-scale and defines the environment size. m_2 discretizes the space further and increases the resolution. Note that usually grid-scales would differ only by factor of about 1.4, but this combination highlights the difference to the combinatorial approach better. (b) The combination of the two grid cell modules forms a grid code. The code is unique across a distance defined by the common multiple of the grid-scales and the number of grid cells per module. Adding another module would greatly increase the coverable environment size. 6
2.4. Firing of a single place cell from the rat brain. Left, the traveling trajectory of the rat is shown in black, spike locations in red. Right, the cell activity is shown as heat map, where red represents high activity and blue low activity. Figure adapted from E. I. Moser, Kropff, et al. (2008). 7

3.1. (a) Detector neuron located at x_j and tuned to detect phase-offsets in the direction of θ_j. It is excited most strongly at a pattern offset of δ. (b) Exemplary network of $3 \times 3 \times 8$ phase-offset detectors, equally spaced across a neuron sheet. Here we distinguish between 8 offset directions only. (c) Current grid cell spiking $\mathbf{s}$ in blue and goal spiking $\mathbf{t}$ in grey. Only three phase-offset detectors spike significantly, as their location x_j is favorable and they are tuned for $\theta_j = 0°$. 11

3.2. Architecture of the system proposed by Banino et al. (2018). It uses a LSTM to simulate grid cell like behavior and another LSTM to decode two grid cell codes into an actionable goal vector. Figure from Banino et al. 2018 12

5.1. Overview of the central part of the system architecture. Several neurological networks are combined to create a cognitive map of the environment. . . . 18
5.2. System overview of the exploration phase where the agent starts creating a cognitive map of the environment. 19
5.3. Neuron sheet of $n \times n$ neurons (blue) where $n = 4$. The East and West, as well as the North and South edges are connected. Light gray arrows indicate the preferred direction $\hat{e}_{\theta_i}$ of each neuron, which is used when calculating the recurrent connectivity profile and when applying an external velocity input to the network. 20
5.4. System overview of the Navigation Phase where the agent uses the cognitive map to navigate to the reward location. 24
5.5. The agent checks for obstacles in 16 allocentric, the goal vector and the heading direction. lookahead directions are considered as blocked if $d_\theta <$ 1.3m or $d_{\theta \pm \alpha} < 0.9$m. The thick arrow is the blocked goal vector, light gray arrows represent blocked directions, all other dark grey arrows non-blocked directions. The dark grey circle marks the distance $d = 1.3$m, the light grey circle a distance of $d = 0.9$m and the orange circle the minimum distance of $d = 0.3$m. 26
5.6. (a) The current grid cell spiking $\hat{S}$ is represented in blue, and the connections C_i between the grid cells and the target place cell i are represented in grey. We are showing four grid cell modules with different velocity scaling factors g_m. The modules with indexes 0 and 2 can detect changes in the x-direction and modules 1 and 3 in the y-direction. The projections are visualized as bar charts right next to the module in the corresponding direction. (b) linear lookahead after $\sim$ 500 time steps in the positive x-direction. Modules 1 and 3 are shown at the moment when the patterns align best. The x-directed place cell firing value of an individual place cell $p_{i|x}$ reaches its maximum here. (c) linear lookahead after $\sim$ 260 time steps in the positive y-direction. Modules 0 and 2 are shown at the moment when the patterns align best. The y-directed place cell firing value of an individual place cell $p_{i|y}$ reaches its maximum here. 29

5.7. The agent encountered an obstacle in the goal vector direction ($d <$ 0.6m). It switches to topology-based navigation and performs a sub-goal localization. It starts a linear lookahead in all non-blocked traveling directions (dark grey thin vectors) and identifies the most promising one (blue vector). 30

6.1. Initialization of grid cell modules loaded before performing the following experiments. Note that the patterns are oriented differently across modules. In both the 2D projection and respective 3D visualization, blue indicates a high spiking value. 33
6.2. Shift in grid cell pattern after applying $\vec{v} = [0.5, 0.5]$ for $k = 2000$ time steps to the grid cell modules. The grey colored pattern represents the initial state, the blue colored one the current state. The spikes are matched via vectors for visualization purposes. Note that the grid cell module with $g_m = 2.4$ seems to be unstable, yet recovers after a short pause. 34
6.3. Experiment setup to test the precision of several grid cell decoding approaches. The robot travels $d = 15$m away from the origin in a random direction. It then computes the goal vector back to the initial position. The absolute distance Δ between actual goal location and estimated location is considered as error. We also considered the deviation in angle α and in length δ between actual and computed goal vector. The experiment is performed 50 times for each decoding mechanism. 35
6.4. Histogram of errors Δ of goal vectors computed with the linear lookahead mechanism from $d = 15$m. Mean average error computes as $\Delta_{avg} = 0.284$m. On the right all trials and their computed goal vector are shown. 36
6.5. Histogram of errors Δ of goal vector computed with the phase-offset detector mechanism from $d = 15$m. Mean average error computes as $\Delta_{avg} = 4.302$m. On the right all trials and their computed goal vector are shown. 37
6.6. Histogram of errors Δ of goal vectors computed with the spike-detector mechanism from $d = 15$m. Mean average error computes as $\Delta_{avg} = 4.877$m. On the right all trials and their computed goal vector are shown. 39
6.7. Maze and agent in pybullet environment. The maze is sized 11×11m and its layout is inspired by Banino et al. 2018. 40

6.8. Cognitive map after hard coded exploration phase of 15000 time steps. Obstacles are colored light gray and doors one to four (medium gray) are still closed. The goal is marked by the blue dot. The circles represent place cells, the connections between them visualize the topology network. The coloring of the place cell is correlated to the reward value of the corresponding reward cell. The darker the blue, the higher is the reward value. 41
6.9. We used the liner lookahead to decode the position vector corresponding to a place cell (right picture). We compared them to the actual position of place cell creation (left plot) and calculated a mean average error of 0.383m. 42
6.10. Examples of sub-goal localizations through performing a linear lookahead in four directions. Direction with highest reward spiking (blue), blocked directions (grey), other non-blocked directions (dark grey). In Figure 6.10a, the dark grey vector corresponds to the actual goal vector. Lookahead data can be found in A.1. 43
6.11. We tested the our proposed algorithm in different maze configurations. The dark grey rectangles are closed doors. The agents trajectory is depicted in blue, the shortest available path is grey dotted. Another trial is shown in Figure 6.12. 45
6.12. Trial 5: only door one is open 47

A.1. Initialization of grid cell modules. The development of their state is shown at several different points in time. In both the 2D projection and respective 3D visualization, blue indicates a high spiking value. 54
A.2. The goal vector computed by the phase-offset detector network has to be scaled with a factor of ρ. We calculated the mean squared error across all 50 trials between the actual distance to the goal and computed vector length. $\rho = 0.285$ minimizes that error. On the right, a single trial is shown, where the agent navigates back to the origin. Distances are adapted to match $\rho = 0.285$. While at the beginning the distance is slightly underestimated, it is overestimated at later times. With our analysis we tried to minimize the errors in both directions, across all trials. 55

A.3. The goal vector computed by the spike-detector mechanism has to be scaled with a factor of ρ. We calculated the mean squared error across all 50 trials between the actual distance to the goal and computed vector length. $\rho = 0.747$ minimizes that error. On the right, a single trial is shown, where the agent navigates back to the origin. Distances are adapted to match $\rho = 0.747$. While at the beginning the distance is slightly underestimated, it is overestimated at later times. With our analysis we tried to minimize the errors in both directions, across all trials. 56
A.4. Histogram of distance Δ between computed and actual goal location at the end of each trial using the spike-detector mechanism. For all trials (left) the mean average error computes as $\Delta_{avg} = 1.276$m. However, if the outliers ($\Delta > 2.5\text{m} \equiv 10\%$ of all trials) are neglected (right), the filtered mean average error becomes $\Delta^*_{avg} = 0.423$m. 56

List of Tables

6.1. Comparison of grid cell decoder mechanisms. The (*) indicates the result where outliers ($\Delta > 2.5$m, $10 - 15\%$ of all trials) were not considered. For the definition of $\Delta, \delta, \alpha, d$ refer to Figure 6.3. 40
6.2. Overview of performance in different maze configurations. The multiples are based on the distance traveled until the goal was first encountered. With known path, we refer to the shortest path in the maze configuration of the exploration phase (only door five open). 47

A.1. Sub-goal localization data for examples shown in Figure 6.10. k is the time-step at which the localization takes place. Only data for non-blocked directions is listed. We determine the highest reward along each direction, list the place cell (pc) firing and at which time step of $dt = 10$ms it occurred. Note that we only reevaluate the reward spiking every 40^{th} step. The look-ahead speed is 0.5m/s and with it we can calculate the distance. Finally, the direction with the highest reward spiking value is chosen. . . . 53

Bibliography

Banino, Andrea et al. (2018). "Representations in Artificial Agents." In: *Nature* 26, p. 1. ISSN: 1476-4687. URL: http://www.nature.com/articles/s41586-018-0102-6%0Ahttp://dx.doi.org/10.1038/s41586-018-0102-6.

Burak, Yoram and Ila R. Fiete (2009). "Accurate path integration in continuous attractor network models of grid cells." In: *PLoS Computational Biology* 5 (2), pp. 1–48. ISSN: 1553734X. DOI: 10.1371/journal.pcbi.1000291.

Burton, Brian G., Vincent Hok, Etienne Save, and Bruno Poucet (2009). "Lesion of the ventral and intermediate hippocampus abolishes anticipatory activity in the medial prefrontal cortex of the rat." In: *Behavioural Brain Research* 199 (2), pp. 222–234. ISSN: 01664328. DOI: 10.1016/j.bbr.2008.11.045.

Bush, Daniel, Caswell Barry, Daniel Manson, and Neil Burgess (2015). "Using Grid Cells for Navigation." In: *Neuron* 87 (3), pp. 507–520. ISSN: 10974199. DOI: 10.1016/j.neuron.2015.07.006.

Edvardsen, Vegard (2015). "A Passive Mechanism for Goal-Directed Navigation using Grid Cells." In: pp. 191–198. DOI: 10.7551/978-0-262-33027-5-ch039.

– (2017). "Long-range navigation by path integration and decoding of grid cells in a neural network." In: *Proceedings of the International Joint Conference on Neural Networks* 2017-May, pp. 4348–4355. DOI: 10.1109/IJCNN.2017.7966406.

Edvardsen, Vegard, Andrej Bicanski, and Neil Burgess (2020). "Navigating with grid and place cells in cluttered environments." In: *Hippocampus* 30 (3), pp. 220–232. ISSN: 10981063. DOI: 10.1002/hipo.23147.

Eichenbaum, Howard (2017). "Memory: Organization and Control." In: *Annual Review of Psychology* 68, pp. 19–45. ISSN: 15452085. DOI: 10.1146/annurev-psych-010416-044131.

Erdem, Uğur M. and Michael Hasselmo (2012). "A goal-directed spatial navigation model using forward trajectory planning based on grid cells." In: *European Journal of Neuroscience* 35 (6), pp. 916–931. ISSN: 0953816X. DOI: 10.1111/j.1460-9568.2012.08015.x.

Fiete, Ila R., Yoram Burak, and Ted Brookings (2008). "What grid cells convey about rat location." In: *Journal of Neuroscience* 28 (27), pp. 6858–6871. ISSN: 02706474. DOI: 10.1523/JNEUROSCI.5684-07.2008.

Hafting, Torkel, Marianne Fyhn, Sturla Molden, May Britt Moser, and Edvard I. Moser (2005). "Microstructure of a spatial map in the entorhinal cortex." In: *Nature* 436 (7052), pp. 801–806. ISSN: 00280836. DOI: 10.1038/nature03721.

Hok, V, E Save, P P Lenck-Santini, and B Poucet (2005). *Coding for spatial goals in the prelimbicinfralimbic area of the rat frontal cortex*. URL: www.pnas.orgcgidoi10.1073pnas.0407332102.

Kropff, Emilio, James E. Carmichael, May Britt Moser, and Edvard I. Moser (2015). "Speed cells in the medial entorhinal cortex." In: *Nature* 523 (7561), pp. 419–424. ISSN: 14764687. DOI: 10.1038/nature14622.

McNaughton, Bruce L., Francesco P. Battaglia, Ole Jensen, Edvard I. Moser, and May Britt Moser (2006). "Path integration and the neural basis of the 'cognitive map'." In: *Nature Reviews Neuroscience* 7 (8), pp. 663–678. ISSN: 1471003X. DOI: 10.1038/nrn1932.

Moser, Edvard I., Emilio Kropff, and May Britt Moser (2008). "Place cells, grid cells, and the brain's spatial representation system." In: *Annual Review of Neuroscience* 31, pp. 69–89. ISSN: 0147006X. DOI: 10.1146/annurev.neuro.31.061307.090723.

Moser, Edvard I., May Britt Moser, and Bruce L. McNaughton (2017). "Spatial representation in the hippocampal formation: A history." In: *Nature Neuroscience* 20 (11), pp. 1448–1464. ISSN: 15461726. DOI: 10.1038/nn.4653.

Moser, May-britt, David C Rowland, and Edvard I Moser (2015). "Place Cells, Grid Cells, and Memory." In: pp. 1–15.

O'keefe, J and D H Conway (1978). *Experimental Brain Research Hippocampal Place Units in the Freely Moving Rat: Why They Fire Where They Fire*, pp. 573–590.

O'Keefe, J and J Dostrovsky (1971). "The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat." In: *Brain Research* 34, pp. 171–175.

Place, Ryan, Anja Farovik, Marco Brockmann, and Howard Eichenbaum (2016). "Bidirectional prefrontal-hippocampal interactions support context-guided memory." In: *Nature Neuroscience* 19 (8), pp. 992–994. ISSN: 15461726. DOI: 10.1038/nn.4327.

Poulter, Steven, Tom Hartley, and Colin Lever (2018). "The Neurobiology of Mammalian Navigation." In: *Current Biology* 28 (17), R1023–R1042. ISSN: 09609822. DOI: 10.1016/j.cub.2018.05.050. URL: https://doi.org/10.1016/j.cub.2018.05.050.

Sargolini, Francesca, Marianne Fyhn, Torkel Hafting, Bruce L Mcnaughton, Menno P Witter, May-Britt Moser, and Edvard I Moser (2006). *Conjunctive Representation of Position, Direction, and Velocity in Entorhinal Cortex*, pp. 758–762.

Solstad, Trygve, Charlotte N. Boccara, Emilio Kropff, May-Britt Moser, and Edvard I. Moser (2008). "Representation of Geometric Borders in the Entorhinal Cortex." In: *Science* 322 (5909), pp. 1861–1865.

Solstad, Trygve, Edvard I. Moser, and Gaute T. Einevoll (2006). "From grid cells to place cells: A mathematical model." In: *Hippocampus* 16 (12), pp. 1026–1031. ISSN: 10509631. DOI: 10.1002/hipo.20244.

Spiers, Hugo J., H. Freyja Olafsdottir, and Colin Lever (2018). "Hippocampal CA1 activity correlated with the distance to the goal and navigation performance." In: *Hippocampus* 28 (9), pp. 644–658. ISSN: 10981063. DOI: 10.1002/hipo.22813.

Stensola, Hanne, Tor Stensola, Trygve Solstad, Kristian FrØland, May Britt Moser, and Edvard I. Moser (2012). "The entorhinal grid map is discretized." In: *Nature* 492 (7427), pp. 72–78. ISSN: 00280836. DOI: 10.1038/nature11649.

Taube, Jeffrey S, Robert U Muller, and James B Ranck (1990a). *Head-Direction Cells Recorded from the Postsubiculum in Freely Moving Rats. I. Description and Quantitative Analysis*, pp. 420–435.

– (1990b). *Head-Direction Cells Recorded from the Postsubiculum in Freely Moving Rats. II. Effects of Environmental Manipulations*, pp. 436–447.

Tejera, Gonzalo, Martin Llofriu, Alejandra Barrera, and Alfredo Weitzenfeld (2018). "Bio-Inspired Robotics: A Spatial Cognition Model integrating Place Cells, Grid Cells and Head Direction Cells." In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 91 (1), pp. 85–99. ISSN: 15730409. DOI: 10.1007/s10846-018-0852-2.

Tolman, Edward C (1948). *THE PSYCHOLOGICAL REVIEW COGNITIVE MAPS IN RATS AND MEN *.*