Transition scale-spaces: A computational theory for the discretized entorhinal cortex

Nicolai Waniek
nicolai.waniek@de.bosch.com
Bosch Center for Artificial Intelligence (BCAI), Robert Bosch GmbH, Robert-Bosch-Campus 1, 71272, Renningen, Germany.

Goal-directed spatial navigation is fundamental for mobile animals and is generally attributed to Place Cells (PCs) and Grid Cells (GCs) of the Hippocampus. It was proposed recently that GCs optimally encode transitions in spatiotemporal sequences. However, a single scale of transitions exhibits unfavorable runtimes when planning long sequences. This issue is approached by observing a relationship to binary search and data structures to optimally accelerate it. Then, this insight is extended to biologically plausible neural representations. The resultant data structure is a scale-space that learns approximate transitions and has an optimal scale-increment of $\sqrt{2}$ between subsequent scales. Also, the results generalize beyond navigation and, thus, describe a general-purpose cortical data structure. Applied to navigation, it resembles multi-resolution path planning, a technique widely deployed in mobile robotics. In addition, the scale-space can be used to find short-cuts, shown in a simulated Morris water maze experiment. Finally, the results provoke a novel understanding of Theta Phase Precession (TPP).

1 Introduction and Motivation

Spatial navigation and localization depend on a number of cells which represent spatial modalities (Rowland et al., 2016). The most perplexing of these are Grid Cells (GCs) of the Entorhinal Cortex (EC) (Hafting et al., 2005). Their eponymous periodic spatial firing patterns, called grid fields, regularly tesselate space. Embedded in an inhibitory local recurrent circuit (Couey et al., 2013), they interact with other spatially modulated cells of the EC, such as head direction (Ranck, 1984; Sargolini et al., 2006), border (Savelli et al., 2008), and speed cells (Kropff et al., 2015). Moreover, they are influenced by the geometry of an animal's surrounding (Krupic et al., 2014; Wernle et al., 2017), and project to PCs of the Hippocampus (Fyhn et al., 2008; E. I. Moser and M. B. Moser, 2008). In turn, PCs – themselves crucial for localization and path planning (Morris et al., 1982) – recurrently project to the EC (Bonnevie et al., 2013).

Early on, theoretical models used grid fields of different sizes to generate place fields (Solstad et al., 2006), the distinct spatial firing

regions of PCs (O'Keefe, 1979). And in fact GCs organize in modules along the dorsoventral axis of the EC (Stensola et al., 2012): Cells within one Grid Module (GM) share the orientation of the hexagonal pattern, show a relative phase shift to each other, and express the same size with respect to their grid fields. Surprisingly though, the sizes of the fields increase from one GM to the next in discrete steps with a scale factor that is suspiciously close to $\sqrt{2}$.

GCs are considered a vehicle to understand higher cortical representations not only due to these remarkable properties (Edvard I. Moser et al., 2014), but also because they are situated at the apex of the cortical processing hierarchy (Felleman and Van Essen, 1991). They are commonly believed to perform either of two functions. On the one hand, hexagonal grid fields were shown to be suitable for path integration (Burak and Fiete, 2009; Fuhs and Touretzky, 2006; McNaughton et al., 2006). On the other hand, they theoretically outperform place cells in localization tasks (Mathis et al., 2012; Stemmler et al., 2015), and were recently shown to emerge in recurrent models for vector-based navigation (Banino et al., 2018; Cueva and X.-X. Wei, 2018). However, the difference between most existing models lies primarily in the way that either of the two is achieved (see reviews in Giocomo et al., 2011; Shipston-Sharman et al., 2016; Zilli, 2012).

Both perspectives raise several concerns. First, many path integration models quickly accumulate noise and therefore require additional mechanisms to prevent drift, especially when deployed in real-world scenarios (Mulas et al., 2016). Second, GCs emerge only after PCs during postnatal development (Langston et al., 2010; Wills et al., 2010), casting doubts on their contribution to localization. Moreover, their contribution to PC activity remains disputed because it is unclear how downstream neurons should read out the ambiguous hexagonal representation (Bush et al., 2014). Third, any vector-based navigation model that uses multiple scales of GCs to represent a target vector generalizes only up to an area that depends on the number of scales. Ambiguous target vectors will appear inevitably as soon as the surrounding environment with which the model has to cope is too large. To conclude, the computational purpose of GCs remains opaque and existing theories lack a satisfying unified explanation for both of their peculiarities, namely their hexagonal grid fields as well as discrete scale increments.

This is the second in a series of papers that addresses these concerns. The series suggests that GCs are part of a fundamental cortical computation in which GCs encode transitions in spatiotemporal sequences, similar to transitions in Reinforcement Learning (RL) or Turing machines. This proposal is evaluated theoretically and with simulations from navigation and path planning. The first paper of the series analyzed the logic of transition encoding in sequences, introduced Multi-Transition Systems (MTS), and proved that a hexagonal distribution of transition

encoders is optimal given an Euclidean input space (Waniek, 2018). The present paper describes why a scale-increment of $\sqrt{2}$ emerges naturally in transition systems to accelerate computations. Furthermore, it speculates about the relationship between scale-space formation in MTS and Theta Phase Precession (TPP), a temporal effect of PCs and GCs during Theta oscillations (Jeewajee et al., 2013). The following papers will provide a biologically plausible model for the self-organization of dendritic computation within a single GC, and present a model for selecting optimal trajectories in MTS. Collectively, the series covers all three levels of computation advocated by David Marr (Marr, 1982), i.e., the computational theory, the representation and algorithms, and the hardware implementation.

Note that this paper stays on the second level of Marr's suggested analysis. Specifically, it uses an algorithm and data structure centric perspective on sequence planning with transition systems to understand and explain observations in the rodent brain as consequences of optimal representations. Hence, the presented results are not implemented in simulations of spiking neural networks. Moreover, self-organization and learning of grid fields and grid scales in such models is omitted. Nevertheless, the enforced algorithmic perspective leads to predictions for real neural networks, which will be discussed in depth.

This paper is organized as follows. First, related work is presented. Then, MTS are briefly reviewed to introduce necessary concepts and nomenclature, making the paper self-contained. Afterwards, a bottleneck in transition systems is highlighted. This is followed by a scale-space model for transition look-ahead to solve this bottleneck. Then, simulations are used to characterize the model. After discussing the obtained result in detail, the requirements it introduces, and comparing it in depth to other existing work, the paper concludes with a summary and a brief outlook to future work.

2 Related work

The periodic firing fields of GCs introduce ambiguities when used for localization. A straightforward and widely adopted solution is to use independent GCs with multiple sizes of grid fields (Solstad et al., 2006). Converging the responses onto a single sheet of neurons, for instance using multiplication or summation, yields unambiguous singly peaked activity in environments that are significantly larger than the biggest grid period (Mathis et al., 2012). A theoretical study used constrained optimization to minimize the number of required neurons for this purpose and showed that this technique is optimal for localization when grid fields increase in discrete steps by a factor of $\sqrt{e}$, or about $1.4 - 1.7$ when neurons fire stochastically (X.-x. Wei et al., 2015). Others proposed that multiple scales of GCs form a population vector for localization that can be read out using Bayesian inference (Stemmler et al., 2015). In this case, the scale ratio was found to be ideally 3/2.

Although these values are in the range of measured scale increments, there exist doubts about the contribution of GCs to localization (Bush et al., 2014).

A functional model which uses several grid scales for trajectory planning is the Hierarchical Look-Ahead Trajectory Model (HiLAM) (Erdem et al., 2015). The model first localizes a target using the largest scale, and subsequently narrows down on the correct location using finer scales. Due to inherent ambiguities when using GCs for localization, the model shows deficiencies when the distance to the target is larger than the largest grid period.

Transition encoding was previously suggested for the Hippocampus, and explored in a biologically plausible model (Cuperlier, Laroque, et al., 2004; Cuperlier, Quoy, et al., 2006). It was evaluated on a robotics platform, and later extended via RL for the selection of an ideal trajectory to a target location (Hirel et al., 2010). The model uses visual processing to extract landmarks, while transition cells support learning and planning of action sequences given visual stimuli. Moreover, it contains detailed biologically plausible hippocampal-prefrontal interactions. However, it neither addresses GCs nor multiple scales of representation.

This work is also related to the idea that GCs perform Principle Component Analysis (PCA)(Dordek et al., 2016), or that they compute a Successor Representation (SR)(Dayan, 1993; Momennejad et al., 2016; Stachenfeld et al., 2016). Although MTS, and in particular spatial transition bundles, are similar to the first in that they compute correlations between states, they are different, among other things, in that they perform the computation directly in their dendritic tree and not due to PC afferents, and learn co-activation with place-cells. The relationship and differences between MTS on the one hand, and PCA and SRs on the other hand will be discussed in further detail in Section 6.

Given that MTS learn transitions in state spaces, this work is complementary to geodesic transition cells described by Gustafson and Daw, 2011, who make accurate predictions about the deformations of the grid code. Spatial transitions from one symbol to another in MTS can be considered locally Euclidean, but the entirety of transitions may actually live on a Riemannian manifold due to topological deformations of the input space. Therefore, computing a shortest path using an MTS corresponds to computing the geodesic and not necessarily to the Euclidean shortest path. In the work by Gustafson and Daw, 2011, hexagonal fields are due to an oscillatory inference model of GCs, whereas hexagonality is an optimality result in MTS. In addition, their work lacks an account of the functional purpose of a multi-scale grid code.

A recent study argued from an information theoretic point of view that neural networks that have to perform optimal planning operations should *in general* express some form of discretized scales (McNamee

et al., 2016). In particular, the discretization was proved to minimize the description length of a planning task and thereby achieve efficient encodings and optimal retrieval. The theoretical investigation did not provide any link to GCs, though. Nevertheless, the study points to general principles of neural computation that involve multiple scales.

Multi-resolution path planning has been an active field of research in robotics for several decades. Many proposed algorithms are based on a fundamental paper by Kambhampati and Davis, 1986, who used a quad-tree to accelerate path planning. Extensions of their idea include trajectory planning in rough terrain (Pai and Reissell, 1995), representations with variable resolutions (Sven Behnke, 2004), dynamic path planning (Lu et al., 2011), or compact mesh representations (Hwang et al., 2003), or beamlets (Lu et al., 2011). These algorithms usually represent an environment topologically or with an imposed grid and use some variant of Dijkstra's algorithm (Dijkstra, 1959) or A* (Hart et al., 1968) to determine an optimal path from start to target.

3 Introduction to Multi-Transition Systems

This section briefly introduces Multi-Transition Systems (MTS) to make the paper self-contained. A detailed treatment, including formal proofs and an exhaustive discussion, can be found in Waniek, 2018.

Consider an animal that wants to plan a path from its current location A to a remote target B. This planning operation requires expansion of the path, denoted $A \rightsquigarrow B$, into a sequence $A, \sigma_1, \sigma_2, \ldots, B$ of $N \geq 0$ intermediate locations σ_i. The animal needs knowledge about such locations for this operation, but also information about feasible transitions between them. Both can be stated in an abstract symbolic framework for sequence learning and generation as follows.

Each location is represented by a symbol σ_i of the set of symbols Σ. Transitions τ_k are typically viewed as functions that map one symbol to another, for instance in RL (Sutton and Barto, 1998). In the context of MTS, they are written either as $\tau_k = (\sigma_m \rightarrow \sigma_n)$ or with the tuple notation $\tau_k = (\sigma_m, \sigma_n)$ to emphasize that they are edges in a graph. To reduce the number of neurons that are necessary to store transitions, MTS use the novel concept of transition bundling. That is, a single neuron can learn not only one transition τ_k, but an entire set of transitions.

Transition bundling is restricted by certain logical constraints. Essentially, only those transitions with mutually exclusive source and target symbols can be bundled. In spaces where arbitrary transitions between any two symbols are possible, for instance in the case of temporal transitions between abstract symbols, a bundle contains only transitions that start at one source symbol. In complete metric spaces, bundling can be performed for all those transitions for which the intersection of the sets of source and target symbols is empty. In a two-dimensional space and given a process that optimally samples symbols from the input space

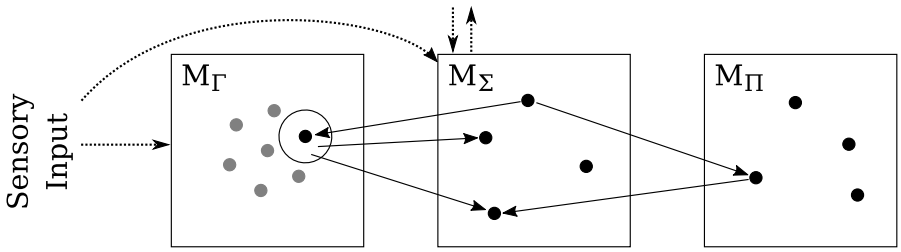

Figure 1: Conceptual overview of the hippocampal-entorhinal Multi-Transition System proposed in (Waniek, 2018). PCs are symbols of neural memory M_Σ and acquired by spatially modulated afferents and other cortical interactions (dotted arrows to and from M_Σ, black dots represent several neurons). They interact with neurons of neural memory M_Π that represent temporal transition bundles (black arrows between M_Σ and M_Π, black dots indicate individual bundles). Spatial transition bundles, stored in memory M_Γ, self-organize by optimally sampling from the sensory input space Δ (dotted arrow to M_Γ). One spatial transition bundle (black dot in M_Γ) exposes several grid fields (gray dots). For more information, see Section 3 or Waniek, 2018.

according to the Peterson-Middleton theorem (Petersen and Middleton, 1962), it turns out that bundling is maximized by a periodic hexagonal grid. In other words, bundles form a dense packing of on-center and off-surround receptive fields, and the symmetry of the packing depends on the structure of the input space. Hence, symmetry is expected to decline when the input space is not uniform in all directions. That is, the grid code of MTS is expected to deform in the way described previously by Gustafson and Daw, 2011.

The hippocampal-entorhinal loop forms an MTS as follows, illustrated in Figure 1. Symbols are represented by PCs in a neural memory M_Σ and recruited from spatial, contextual, and possibly other cortical afferents. Temporal transitions between these symbols are bundled in a neural memory M_Π and allow reproduction of exact temporal sequences. Without other afferents, any neuron in M_Π will co-activate with the PC of M_Σ for which it defines feasible temporal transitions, effectively inheriting the PC's place field. Finally, spatial transitions are learned by GCs of neural memory M_Γ. In contrast to temporal transitions, GCs perform optimal sampling of a spatially modulated input space as part of their dendritic computation. This discretization of the input space is proposed to be performed by dendritic computations, and generates a symbolic input space $\delta_j \in \Delta$ that is part of the dendritic tree of spatial transition bundles. Each bundle learns co-activation with symbols $\sigma_k \in M_\Sigma$ to recurrently convey relational information about spatially neighboring PCs $\sigma_l, l \neq k$. Moreover, each transition bundle exposes multiple receptive fields, each of which corresponds to an individual transition.

Due to co-activation learning of PCs and recurrent projections to M_Σ, GCs learn spatial neighborhood relationships between PCs. Specifically, they learn feasible spatial transitions from PCs that are active at a

location to all PCs in the surrounding region. To summarize, neurons of M_Σ form nodes of a topological map, while neurons of M_Γ and M_Π represent edges between nodes. Figure 1 shows strong coupling between memories M_Σ and M_Π, which can be identified with Cornu Ammonis 3 (CA3) and Cornu Ammonis 1 (CA1). This prediction of a strong dependence of CA1 on CA3 was recently supported by novel evidence from rodent experiments (Davoudi and Foster, 2019).

Decoupling spatial transition knowledge from PCs via GCs has several benefits. Most importantly, it renders symbols independent from the structure of sensory input. Although spatial information is required during learning of grid fields and sequences, sensory input should not be necessary during retrieval and planning of a sequence to a desired symbol. Also, a currently active representation of PCs may change abruptly, for instance due to a contextual change that induces a remapping event (Muller and Kubie, 1987). Metric knowledge about feasible movements remains intact within GCs during such an event, because GCs are anchored to the sensory space independently from PCs as part of their dendritic computation An in depth discussion about this and several other benefits can be found in Waniek, 2018. Recently, also others noted the functional benefits of separating (i) transitions that are recruited from sensory inputs from (ii) spatial representations that live independently of the structure of the sensory input, and provided an elegant probabilistic account of their model (Whittington et al., 2018).

4 Scale-spaces for optimal transition look-ahead

What are the computational implications of transition encoding during retrieval and how do they relate to behavioral necessities? This section tackles the question and its arising problems with an algorithm and data structure centric perspective and with examples from spatial navigation. However, the results extend beyond navigation and apply to other transition systems. In particular, temporal transition systems for episodic memories are expected to form scale-space structures along appropriate dimensions similarly to the spatial structure described in the remainder of this section.

This section presents an optimal solution to undesirable run-times in MTS and, generally speaking, model-based algorithms. Note, however, that biologically plausibly activating and monitoring specific symbols in M_Σ, i.e., activating the starting symbol and waiting until the target symbol becomes active, or waiting a certain maximal run-time are not subject of this paper. They are expected to be the result of prefrontal-hippocampal interactions, though. Also, selection of an optimal solution in the case of multiple feasible trajectories from A to B is not addressed in a biologically plausible manner or with methods from RL, but with techniques from graph algorithms. However, augmenting the solution with reward propagating appears to be straightforward. These points will be discussed in further detail in Section 6.

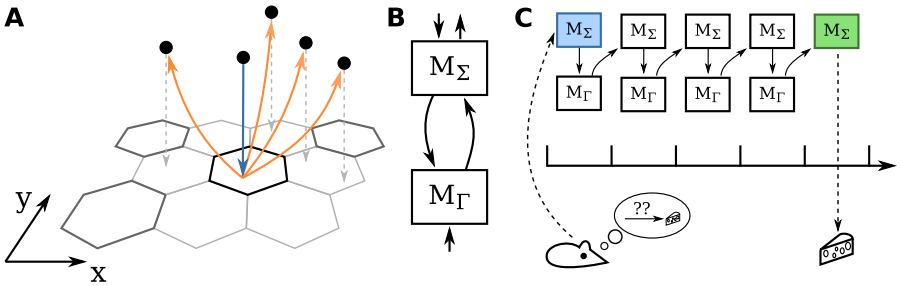

Figure 2: (A) Discrete Voronoi clustering due to absolute WTA dynamics and association with symbols in an idealized space. Given a flat input space (indicated by the x-y axes), Voronoi clustering by cells in M_Γ yields hexagonal fields. A single GC of M_Γ (black hexagon, periodically repeating on-center fields indicated by dark gray hexagons) associates with all afferent symbols of M_Σ that are within its *on-center* regions (black dot with blue arrow pointing towards black hexagon) due to co-activation learning during exploration, and learns transitions (orange arrows) to other symbols (black dots with gray dashed arrows) in their *target* or *off-surround* region (light gray hexagons) of the transition. (B) Simplified modular architecture of Figure 1, including recursive loop and external interactions, and shown here only for M_Σ and M_Γ. (C) Given activation of a starting symbol (indicated by blue color of M_Γ) on a linear track that was discretized into bins (black arrow with vertical bars), mentally travelling to a target involves recursively activating M_Σ and the transition memories M_Π and M_Γ (only M_Γ shown here) until the target symbol is active in M_Σ (indicated by a green box).

4.1 Run-time complexity of pre-play activity

Imagine an animal that is equipped with a neural MTS $\mathcal{M}$ that it uses for navigational purposes. That is, it uses a finite number of neurons to represent locations via symbols of a memory M_Σ and movements between two symbols with spatial and temporal transition bundles of memories M_Γ and M_Π, respectively. Here, $\mathcal{M}$ is subject to several simplifications. In particular, any uncertainty in the perceptual representation of the input space is ignored and spatial transitions are assumed to organize according to a Winner-Take-All (WTA) mechanism. Then, the fields of all spatial transition bundles perform a Voronoi tesselation, which – given an idealized two-dimensional input space that is uniform along all dimensions – forms hexagonal fields. These simplifications will be relaxed in Subsection 4.3, which considers biologically plausible receptive fields.

The modular organization of memories M_Σ, M_Γ and M_Π is depicted in Figure 1. Figure 2A illustrates the hexagonal tesselation and how symbols are associated with spatial transitions. Figure 2B emphasizes the recurrent connectivity exemplary for M_Σ and M_Γ.

Assume that the animal explored an environment sufficiently long and learned symbols and transitions accordingly. Now, it wants to know if it can go from its current location $A \in \Sigma$ to a remote food source $B \in \Sigma$. The animal can answer this question by mentally traveling from A to B, essentially pre-playing the places it will visit. In terms of MTS,

it has to expand the path $A \rightsquigarrow B$ into a sequence $A, \ldots, \sigma_i, \ldots, B$ of intermediate symbols σ_i. This can be achieved algorithmically by first activating A in M_Σ and then (indirectly) recursively looping through M_Σ via the memories M_Π and M_Γ until B activates in M_Σ, illustrated in Figure 2C for the interactions between M_Σ and M_Γ, or until some maximal running time is exceeded. The animal will then know if there exists (at least one) sequence by which it can reach its destination. Note that recursive retrievals via M_Π and M_Γ coincide on a linear track. The runtime complexity of this operation resides in $O(N)$. That is, it depends linearly on the number of recursive evaluations.

The linear dependency on N is an issue especially when temporal dynamics of real neural networks are respected. Consider an implementation of $\mathcal{M}$ that consists of neural associative memories for M_Σ, M_Γ, and M_Π, and with temporal dynamics that consume about 10 ms per memory for spike generation, propagation, and axonal delays. Furthermore, let A and B be on a linear track with distance $D = 200\,\text{m}$ and grid field size $d_g = 20\,\text{cm}$. As above, recursive expansions through M_Γ and M_Π coincide due to linearity of the track. The expanded path from A to B consists of 1000 intermediate invocations of the $M_\Sigma - M_\Gamma$ loop. In total, this consumes 20 s for mentally travelling from A to B via recursive expansion, during which the animal may easily fall victim to a roaming predator. Figure 2C shows an illustration of the recursive retrieval using memories M_Σ and M_Γ. The runtime will increase further when there is a set of candidate sequences from which the animal has to chose, for instance with some reward propagation mechanism or graph search, and especially with stochastic neural responses. Moreover, the expansion must be repeated for each novel target. This behaviorally questionable performance for model-based algorithms was previously noted in the context of GCs by Dordek et al., 2016 and Stachenfeld et al., 2016. Summarizing, recursive retrieval in a single-scale transition system has a time complexity that is impractical from a behavioral perspective.

4.2 Optimal searches on discrete data

The information stored in $\mathcal{M}$ forms linked lists of temporally and spatially ordered elements. Consider the worst case of a linear track, as introduced above. Then, a query about the *existence* of a path from A to B corresponds to searching an element in a linear linked list. Such queries can be accelerated optimally with the help of well-known and widely deployed data structures. Particularly interesting ones in the context of transition encoding will be briefly introduced here.

Skip lists form a hierarchy of *fast lanes* that jump over elements of the original list (Pugh, 1990). Searching an element in a skip list starts on the highest (or coarsest) level of the hierarchy and proceeds until an element is found that is larger than the target. Subsequently, the search drops down to the next finer level of the hierarchy until

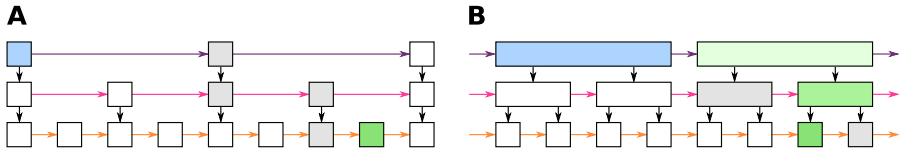

Figure 3: (A) A skip list is a multi-layer linked list data structure (Pugh, 1990). Elements of the list must be ordered. Higher layers have fast lanes to skip over elements. Given a start (blue), searching an element (green) in a skip list starts in the highest layer. Skip links are followed until an element on the corresponding layer is found which is larger than the target. Then, the search is continued on a lower level (see gray boxes). (B) In contrast to skip lists, interval skip lists (Hanson, 1991) store intervals in higher layers. Searching an element is similar to skip lists, but tests if the target element is in an interval (indicated by larger boxes), and subsequently drops down in the hierarchy. Interval skip lists are related to segment trees (see de Berg et al., 1997).

the element is recovered. Given uniformly sampled data, skip lists are dual to binary search (Dean and Jones, 2007) and thus become asymptotically optimal (Knuth, 1998). Specifically, the hierarchy of fast-lanes improves operations on equidistant data exponentially. That is, retrievals, insertions, and deletions can be performed in $O(\log N)$ time with high probability. Due to their favourable properties, they found widespread application in various applications (for instance see Bender et al., 2017 and references therein). In addition, skip lists generalize to higher dimensions (Eppstein et al., 2005). Searching in a skip list is illustrated in Figure 4A.

For meaningful behavior, an animal should not only *find* a certain element in its accumulated knowledge. Expansion of sequences from arbitrary start to goal symbols requires to retrieve all relevant intermediate symbols or approximations thereof. This can be computed with *interval queries*. Given a suitable data structure $\mathcal{D}$, for instance an array of elements, an interval query $iq(\mathcal{D}, q_s, q_e)$ returns all intervals in $\mathcal{D}$ that are between query start q_s and query end q_e. An element of a data structure $\mathcal{D}$ can be identified with a symbol σ_i or spatial symbol δ_j in $\mathcal{M}$. Similar to binary search, interval queries can be performed in logarithmic time with suitable data structures, such as segment trees (de Berg et al., 1997), or interval skip lists (Hanson, 1991). Figure 3B depicts searching an element using an interval skip list. Answering $iq(\mathcal{D}, q_s, q_e)$ is performed by first locating the interval that contains q_s. Then, intermediate intervals are retrieved until the interval that contains q_e is found. This is typically performed on the highest available level for performance reasons. Intermediate low-level values can be extracted by traversing the lower levels of the data structure on the chain of higher-level intervals. Thus, reporting all intermediate K values between q_s and q_e can be performed in $O(logN + K)$ time.

Segment trees or interval skip lists can be built bottom up in $O(N)$ time (de Berg et al., 1997). Each additional layer in the tree or list contains nodes with intervals that enclose the nodes of the lower

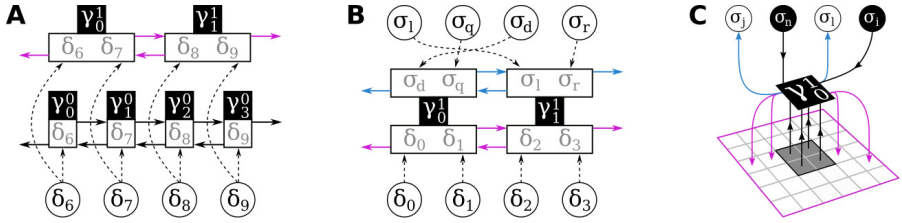

Figure 4: Scale-space construction and co-activation learning in discrete spaces. (A) Construction of an interval skip list for the representation of transitions on symbols $\delta_i \in \Delta$, $i \in N_\Delta$. (B) Construction of scale 1 on symbols $\delta_i in \Delta$, including learning transition on symbols $\sigma_k \in Sigma, k \in N_\Sigma$, whereby structural information from Δ is relayed to Σ. (C) Two dimensional example that emphasizes the on-center (gray shaded area, bottom row) and off-surround (magenta area excluding gray shaded region, bottom row) receptive field characteristics of a transition (black box) and co-activation learning symbols σ (top row: co-active symbols shaded black, transition leads other symbols via blue arrows). For more details, see Subsection 4.2.

level, illustrated in Figure 3B. Likewise binary trees, segment trees and interval skip lists generalize to higher dimensions.

The construction procedure for interval skip lists directly translates to an optimal discrete scale-space MTS $\mathcal{M}^{\mathsf{S}}_{\text{discrete}'}$, in the following presented in terms of spatial transitions. Recall that optimal sampling is part of the proposed dendritic computations of spatial transition bundles to maximally cover their input space. Hence the assumption of uniformly sampled data is well reflected in the axiomatic description and construction of an MTS. Let scale 0 be the lowest level of $\mathcal{M}^{\mathsf{S}}_{\text{discrete}'}$ and denote the set of symbols for which a transition is defined as its *domain* and the set of symbols to which a transition leads to as its *image*. On scale 0, the domain of a spatial transition γ^0_i corresponds to a spatial input symbol δ_i. Its image is given by symbols surrounding δ_i, for instance δ_{i-1} and δ_{i+1} in a one-dimensional setting. The domain of a spatial transition $\gamma^{(n)}_i$ on scale $n > 0$ corresponds to the integrated domains of 2 adjacent transitions of the previous scale along each dimension, where 2 follows from constructing an optimal search data structure (de Berg et al., 1997; Knuth, 1998; Pugh, 1990). Likewise, the image of $\gamma^{(n)}_i$ grows by a factor of 2 along each dimension with respect to scale $n-1$ and encloses all symbols that surround the domain of $\gamma^{(n)}_i$.

Figure 4A illustrates the construction process for a one-dimensional example. The domain of transitions γ^0_i on scale 0 (middle row) corresponds directly to elements of the spatial input space $\delta_k \in \Delta$ (bottom row). To form an additional scale (top row), transitions γ^1_j have domains that are twice as large per dimension as the previous scale. Recall that spatial transition bundles form receptive fields directly on the input space (indicated by dashed arrows). This pooling operation is

suggested to be a dendritic computation (contribution of a symbol to a receptive field is indicated as gray symbol).

Figure 4B depicts construction and simultaneous co-activation learning of two sets of symbols. By learning co-activation of symbols from two different sets, i.e., $\delta_k \in \Delta$ and $\sigma_h \in \Sigma$, transitions learn relationships in both spaces simultaneously. Thereby, transitions convey the structure from one set of symbols to the other. Shown in the figure are encoders γ_i^1 of scale 1 that learn transitions between pairs of symbols in both spaces. In the figure, black boxes represent transition encoders, white boxes their respective dendritic inputs, and circles symbols. Learning the transitions across larger input spaces than singular entities requires a suitable symbol buffer mechanism, which will be discussed in Section 6.

Note that although Figure 4A and B show arrows between the dendritic trees of transition encoders, for instance between γ_0^1 and γ_1^1 in the case of panel B, it is not implied that transitions learn these connections directly. Rather, they learn to transit to the corresponding symbols in the respective symbolic spaces. These indirections were removed for illustrative purposes. Nevertheless, if these direct connections between transition encoders and their represented states were realized in a neural network, they would lead to layer-specific local recurrent dynamics.

Finally, Figure 4C is an illustration of the construction and co-activation learning for a transition of scale 1 in a two-dimensional example. The illustration emphasizes the on-center (gray shaded area, bottom row) and off-surround (magenta area excluding gray shaded region, bottom row) receptive field characteristics of a transition (black box). Transitions start from spatial inputs $\delta \in \Delta$ that are in the on-center region, indicated by black arrows starting in the gray area, and lead to any symbol in the off-surround area, highlighted by magenta arrows and the magenta box. Co-activation learning with other symbols $\sigma_h \in \Sigma$ for $h \in N_h$ happens alike (top row: co-active symbols shaded black, symbols to which the transition leads are white). However, the metric structure of relationships between symbols $\sigma_h, \sigma_l \in \Sigma$ is given only due to the formation of receptive fields on top of Δ.

A simplified illustration for the two-dimensional case that omits any spatial input symbol δ and plots only on-centers of receptive fields of transitions γ is presented in Figure 5A.

4.3 Optimal search under uncertainty

How do segment trees and interval skip lists apply to an MTS $\mathcal{M}^{\mathsf{S}}_{\text{prob}}$ with realistic neural responses? Neurons typically express bell-shaped tuning curves (Butts and Goldman, 2006; Jazayeri and Movshon, 2006), which is why their responses can be interpreted in terms of probability theory. Therefore, let two adjacent spatial symbols be represented by two random variables X_i and X_{i+1} and their corresponding Probability

Density Functions (pdfs) $f_i(x_i)$ and $f_{i+1}(x_{i+1})$. In particular, let X_i be normally distributed with mean μ_i and n-dimensional covariance $\mathbf{\Sigma}_{\text{eigen}}$, i.e.,

$$X_i \sim \mathcal{N}(\mu_i, \mathbf{\Sigma}_{\text{eigen}}) \tag{1}$$

for $i \in \mathbb{N}$. Here, $\mathbf{\Sigma}_{\text{eigen}}$ is expected to depend on the finite number of presynaptic neurons as well as the local (dendritic) decoding scheme. That is, it adheres to the Cramér-Rao bound which presents a lower bound with which a decoding scheme can extract information from pre-synaptic inputs (Cover and Thomas, 2006; Pilarski and Pokora, 2015), and is assumed constant across neurons. Put differently, there is a minimal resolution to discriminate two (sensorily) adjacent positions. In the absence of any other afferents, a neural $\mathcal{M}^{\mathsf{S}}_{\text{prob}}$ can thus only differentiate between symbols with an accuracy that depends on $\mathbf{\Sigma}_{\text{eigen}}$.

Due to optimal construction, described in Subsection 4.2, a transition on scale $n = 1$ of $\mathcal{M}^{\mathsf{S}}_{\text{prob}}$ has a domain that merges the representations of scale 0. Then, the receptive field of its domain is given by a random variable

$$Y = X_i + X_{i+1} \tag{2}$$

with the corresponding pdf defined by

$$g(y) = f_i(x_i) * f_{i+1}(x_{i+1}), \tag{3}$$

where $*$ denotes convolution. Using standard results from probability theory, the convolution of two normally distributed pdfs is another normally distributed pdf with combined mean and covariance. Specifically,

$$Y \sim \mathcal{N}(\mu_j, \Sigma_j) = \mathcal{N}(\mu_i + \mu_{i+1}, 2\mathbf{\Sigma}_{\text{eigen}}). \tag{4}$$

Thereby the integration area of a spatial sampling process *doubles* during the construction of an additional scale in $\mathcal{M}^{\mathsf{S}}_{\text{prob}}$. Also due to construction, μ_j is required to be centered exactly between μ_i and μ_{i+1}, i.e., $\mu_j \triangleq (\mu_i + \mu_{i+1})/2$. The results directly apply to higher dimensions, and can be recursively applied to form arbitrary numbers of scales.

Conclusively, the radius r_{sampling} of receptive fields of the spatial sampling process, which is part of the dendritic computation of each transition bundle, ideally increases by a factor of $\sqrt{2}$ from one scale to the next in $\mathcal{M}^{\mathsf{S}}_{\text{prob}}$. Likewise, the spatial period of transition bundles increases accordingly. Repeatedly applied, the constructive procedure forms a scale-space of transition bundles that learn approximate transitions or, in other words, transition look-aheads. Computing approximate transitions on higher scales essentially blurs the input space and, thus, removes high frequency components. Specifically, the receptive fields of each scale are characterized by $\Sigma^{(n+1)} = 2\Sigma^{(n)}$ and $r^{(n+1)}_{\text{sampling}} = \sqrt{2} r^{(n)}_{\text{sampling}}$, where n denotes the scale and $\Sigma^{(0)}$ is given by the Cramér-Rao bound, i.e., $\Sigma^{(0)} = \mathbf{\Sigma}_{\text{eigen}}$. Figure 5B and C illustrate these results for the discrete and probabilistic case in two dimensions.

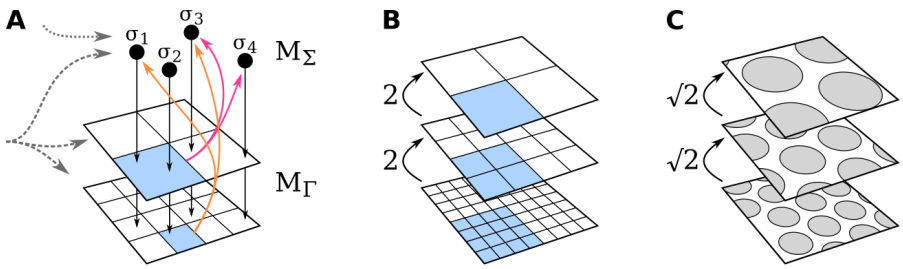

Figure 5: (A) Simplified illustration of learning transitions in a scale-space. Inputs to the scale-space MTS $\mathcal{M}^{\mathsf{S}}_{\text{discrete}}$ are indicated as gray dashed arrows. Transitions of a memory M_Γ learn their domains on the input space according to their scale (blue boxes), and relay this structural information to symbols $\sigma \in \Sigma$ of memory M_Σ (black dots). In this example, a transition that is defined for σ_2 leads to σ_1 and σ_3 on the lowest scale (orange arrows), whereas it leads to σ_3 and σ_4 on scale 1 (magenta arrows). (B) In the discrete case, the optimal scale increment of the domain (receptive field) of transitions is 2 along each dimension. In two-dimensions, illustrated here, this leads to the well known quad-tree data structure. (C) In stochastic input spaces that can be described with normally distributed pdfs, the optimal scale increment from one scale to the next is $\sqrt{2}$.

4.4 Sequence planning and shortcut discovery with transition scale-spaces

This section answers the question how an animal can produce sequences between two arbitrary symbols σ_s and σ_t using a scale-space representation $\mathcal{M}^{\mathsf{S}}$. Note that the presented multi-resolution approaches are similar to a method that was previously described by Sven Behnke, 2004 in the domain of robotic navigation. Moreover, the described methods are graph search algorithms and based on well-known techniques such as Dijkstra's algorithm or the flood fill algorithm. In fact, these algorithms needed only mild extensions to operate on sets and the scale-space structure.

As in the previous section, transitions are treated irrespective of the transition bundle to which they belong. This simplifies the description without violation of the logical considerations of MTS and, thus, improves clarity of the presentation. To further reduce complexity, consider an $\mathcal{M}^{\mathsf{S}}$ without the temporal transition memory M_Π (see Figure 1 for reference). Specifically, $\mathcal{M}^{\mathsf{S}}$ consists of memory M_Σ that stores symbols σ_k, $k \in N_\Sigma$ and memory M_Γ which is a transition scale-space as described in the previous section. Finally, assume that the memories already formed, i.e., the scale-space was already built and corresponding symbols assigned to transition images and domains on each scale.

Consider an $\mathcal{M}^{\mathsf{S}}$ with only a single scale. Finding a sequence from a symbol σ_s to σ_t requires only few operations. First, σ_s must be activated in M_Σ. Then, a corresponding transition γ_i that is defined for σ_s will be activated in M_Γ. In turn, γ_i activates a set $\{\sigma_j\}$ of subsequent symbols σ_j to which γ_i leads to. The combination of these two steps is called

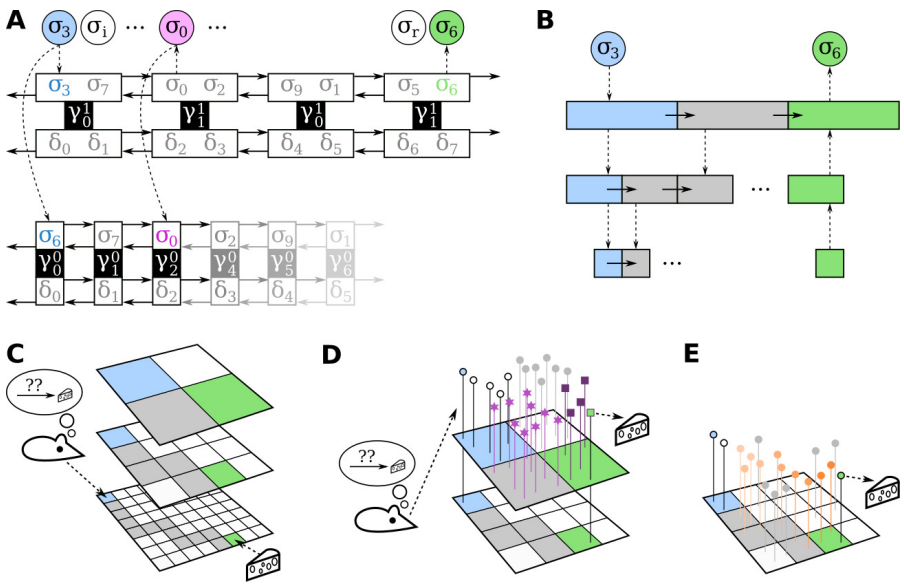

Figure 6: Retrieval of sequences with a scale-space in descending mode. (A) Transitions $\gamma_i^{(n)}$ learn transitions in two spaces simultaneously, where higher scales integrate larger sets of symbols. During retrieval, only one of these spaces is necessary and expansion starts on the highest level from start (blue circle) to goal (green circle). After an approximate solution was found on this level, subgoals are generated (magenta circle). The next lower scale only needs to compute expansion to the target symbol. (B) Simplified depiction of (A) that ignores details on the spatial input space of transitions. (C) An animal wants to compute a path from its current location to a target using a transition scale-space. (D) It can first compute an estimate on the highest scale, prune unnecessary symbols (gray circles), and generate subgoals (magenta stars). (E) The trajectory is subsequently refined by planning on a finer scale to these subgoals.

expansion. If there exists at least one path from σ_s to σ_t, recursively repeating this process will eventually activate σ_t in M_Σ. Obviously, this is the flood-fill algorithm applied on a graph, where nodes correspond to symbols σ_i, and transitions are made explicit by transition encoders γ_j.

During the expansion step, any expanded symbol needs to remember its parent symbols. For instance, if symbol σ_6 activated transition encoder γ_0, which in turn activated σ_7, then σ_7 needs to remember σ_6 as its parent symbol. Then, after successful activation of the target symbol σ_t, a viable sequence from σ_s to σ_t can be found by backtracking from σ_t via any of the parent symbols. This is the backtracking step of Dijkstra's algorithm, modified to select an arbitrary parent from the set of feasible parents. Note that by following the symbolic sequence $\sigma_s \rightsquigarrow \sigma_t$, the spatial sequence can be inferred because transition encoders γ_i learned transitions in the symbolic space Σ and in the spatial input space Δ jointly.

The animal can use multiple scales in several ways. One possibility is to apply the expansion and backtracking algorithms only on a

behaviorally necessary level. For instance, if the animal needs to find an exact sequence, then it should use the lowest level of M_Γ because it contains the highest resolution. If, however, the animal needs only an approximate direction towards a goal or when computational time is limited, it can operate on the highest available level. Intuitively speaking, the highest level represents approximate knowledge and, thus, can be used to find coarse estimates. Another possibility is to use all scales during sequence planning, which can be implemented in different ways.

One variant is *descending mode*, in which the sequence is first approximated on the highest available scale. After finding the target symbol σ_t on the highest scale, backtracking to the starting symbol σ_s yields subgoals. These subgoals are defined as all those symbols that are reachable via only one transition on this scale. Then, the subgoals are used on the next smaller scale to refine the trajectory, illustrated in Figure 6. In both panels of the figure, the start symbol is σ_3 (blue color) and the target is σ_6 (green color). Panel A shows two scales in a one dimensional example and includes all details about the transition encoders $\gamma_i^{(n)}$, where n is the scale. That is, a transition encoder $\gamma_i^{(n)}$ (black box) learns transitions simultaneously in the symbolic space Σ as well as the spatial input space Γ, indicated by white boxes containing corresponding symbols. The expansion and backtracking steps on scale 1 yield intermediate symbols, shown demonstratively only for σ_0 (pink color). Then, the sequence is refined via scale 0 that computes only the trajectory from σ_3 to the sub-goal σ_0. Note that during retrieval, transition encoders $\gamma_i^{(n)}$ don't need access to spatial symbols δ_j. However, these spatial symbol can be inferred due to activation of the transition encoders. Panel B introduces a reduced graphical depiction of this process, in which details about transition encoders are omitted. The panel also illustrates that the integration areas of higher scales don't need to perfectly align at the borders with smaller scales. Figure 6C-E present an illustration of this process for a two-dimensional example in a grid-world. An animal that wants to compute a trajectory from its current location to a target (cheese) first retrieves a coarse estimate that activates multiple symbols in a coarse target region (panel D, purple small squares) and intermediate sub-goals (pink stars). Backtracking effectively reduces the search space for the refinement process. That is, symbols that are not on the coarse path (panel D, gray circles) don't need to be expanded on the next finer scale (panel E).

Another variant is *ascending mode*, in which the search for σ_t starts on the lowest scale. After a number of failed attempts to discover σ_t on this level, the search continues on a larger scale. This is repeated until the target is found, which may eventually, but not necessarily, involve the highest scale. In other words, this automatically adapts the scale to the real distance to the target. In contrast to *descending mode*, this variant requires the backtracking operation only once the target is found.

In both modes, the animal has access to the first steps of a detailed sequence after only a brief computational time, and can refine this sequence while already moving. Conclusively, this solves the runtime issue that was used to motivate the scale-space structure.

Learning multiple scales and associations in a biologically plausible manner, as well as the feasibility of backtracking in real neural networks will be discussed in Section 6. Pseudo-code and further details of the algorithms and sub-routines can be found in Appendix A.

5 Experiments

The scale-space data structure and algorithms were examined using simulations of a virtual agent on a plane surface. In particular, the $\mathcal{M}^{\mathsf{S}}$ that was simulated consists of a memory M_Σ that stores symbols σ_k, $k \in N_\Sigma$ and a memory M_Γ of spatial transitions. An explicit temporal memory M_Π was ignored to keep the implementation lean and comprehensible. If not stated otherwise, symbols and domains and images of transitions were computed in an offline manner. That is, transition centers were distributed hexagonally at the beginning of a simulation. This means that the input space was discretized due to Voronoi clustering; an implementation with a probabilistic representation of transition activations and transition probabilities will be left for future work. In addition, symbols were distributed randomly in the environment. Each transition and symbol was associated with its two dimensional coordinate on the plane. Subsequently, images and domains of transitions were computed from the relative coordinate locations of symbols and transitions. Specifically, each transition pools all symbols according to a nearest neighbor search, and leads to all symbols in its image simultaneously (see Figure 2 for reference).

The source code to reproduce all data for the figures and results presented in this section as well as additional results are available online under an Open Source License[1].

5.1 Worst case timings on a linear track

Segment trees or interval skip lists are widely used and their characteristics well known. The objective of this experiment was thus to simply show the acceleration capabilities of the data structure to find an element in the worst-case scenario.

A simulated agent was put on a linear track of length 10 m to compute the number of recursive retrievals that are required to find a specific element. Spatial symbols were distributed on the track quasi-randomly using the Hammersley point set, a low-discrepancy sequence. Then, a transition scale-space consisting of 7 scales with periods 0.2, $\sqrt{2} \cdot 0.2$, 0.4, $\sqrt{2} \cdot 0.4$, 0.8, $\sqrt{2} \cdot 0.8$ and 1.6 m between transition encoders was computed offline by associating symbols to transition

1 https://github.com/rochus/scalespacemts

Grid Period	0.2	$\sqrt{2} \cdot 0.2$	0.4	$\sqrt{2} \cdot 0.4$	0.8	$\sqrt{2} \cdot 0.8$	1.6
Recursions	50	35	25	18	12	9	6

Table 1: Number of required recursive expansions until a target symbol σ_t is found on a linear track.

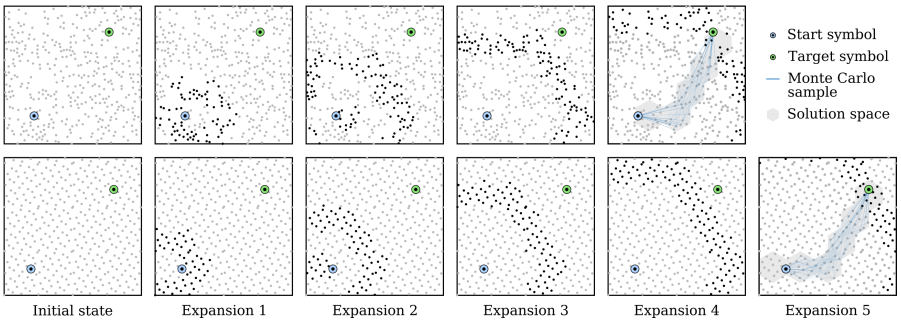

Figure 7: Propagating wave during expansion from start to target in a square environment. Top row shows results for randomly distributed symbols, bottom row for symbol coordinates according to the Hammersley point set. Transition encoders were placed regularly on a hexagonal lattice (not shown). Final panels include the solution space and 300 Monte Carlo samples.

encoders using a nearest-neighbor search. The starting symbol closest to the start of the track was designated as σ_s, and the symbol closest to the end as σ_t. The numbers of necessary recursive retrievals from σ_s until σ_t was found are presented for each scale numerically in Table 1.

5.2 Wave propagation dynamics in two dimensions

In two dimensional settings, the expansion procedure is expected to resemble a propagating wave of activity when no additional restrictions on the expansion are imposed. To examine and illustrate this behavior, expansion was computed according to Algorithm 2 in a square environment of size 1×1 m. A total of 400 spatial symbols were distributed either randomly with a minimal distance of 0.02 m or according to the Hammersley point set. The objective of the latter was to observe any difference in behavior with more regularly placed symbols. Then, transitions were computed for a period of 0.2 m between transition encoders offline. The expansion was started at symbol σ_s, which was defined to be the closest symbol to position $(0.2, 0.2)$, and stopped at the target symbol σ_t, closest to position $(0.8, 0.8)$. After expanding and localizing σ_t, 300 Monte Carlo samples of symbolic sequences were computed using backtracking and randomly selecting a parent during each backtracking step.

Figure 7 shows the results of the expansion and the emerging propagating wave from σ_s to σ_t. Black dots indicate symbols which are currently active during the expansion. The final expansion panel of each row shows a solution space as gray hexagons and Monte Carlo

samples as blue lines. The solution space is given by the sequence of spatial domains of subsequently activated transitions on a solution sequence and forms a channel from start to target.

The only distinctive qualitative difference between randomly and more regularly placed symbols is the form of the propagating wave. When the symbols are regularly placed, then the wavefront is hexadirectional during later expansion steps due to the perfectly regular placement of transition centers. For randomly placed symbols, the wavefront is more tattered, but still resembles the hexadirectional characteristic of the regular case. The difference in the solution space and the number of expansions is due to the selection of the start and target symbols and, thus, negligible.

Note that more than just one solution space may exist due to the underlying assumption of MTS. Specifically, MTS assume a constant-cost operation to move from the domain of a transition to its image. However, only one solution space is illustrated for each example in Figure 7. Also note the active symbols close to the start symbol σ_s during the second expansion. This behavior is because these symbols were not co-activated during start even though they are in close vicinity to σ_s, and because the implemented transition system does not include any knowledge about spatial directivity of the expansion.

5.3 Scale-space planning with sub-goal generation in descending mode

The descending mode of trajectory refinement was analyzed using a rectangular area of $2 \times 5\,\mathrm{m}^2$. In this area, 250 symbols were distributed randomly, keeping a minimal distance of 0.05 m between each pair of symbols A transition scale-space consisting of 5 scales with periods $0.2, \sqrt{2} \cdot 0.2, 0.4, \sqrt{2} \cdot 0.4$ and 0.8 m between transition encoders was then computed offline.

The symbol closest to coordinate $(0.4, 0.1)$ was selected as start symbol σ_s, and the symbol closest to coordinate $(1.8, 4.8)$ as target symbol σ_t. Qualitative results are illustrated in Figure 8. The search starts on the highest available scale, which computes an entire approximate solution from start to target. The solution space of this scale is indicated by gray hexagons in the leftmost panel of the figure. Also, this generates sub-goals, indicated by magenta circles.

After defining subgoals, the procedure drops down to the next scale and computes trajectories only to the previously determined subgoals. This limits the search space on smaller scales. The process repeats until the smallest scale is found, which is illustrated in the other panels of the figure.

5.4 Scale-space planning in ascending mode

The qualitative behavior of the ascending mode was examined likewise the descending variant. That is, 250 symbols were placed randomly in

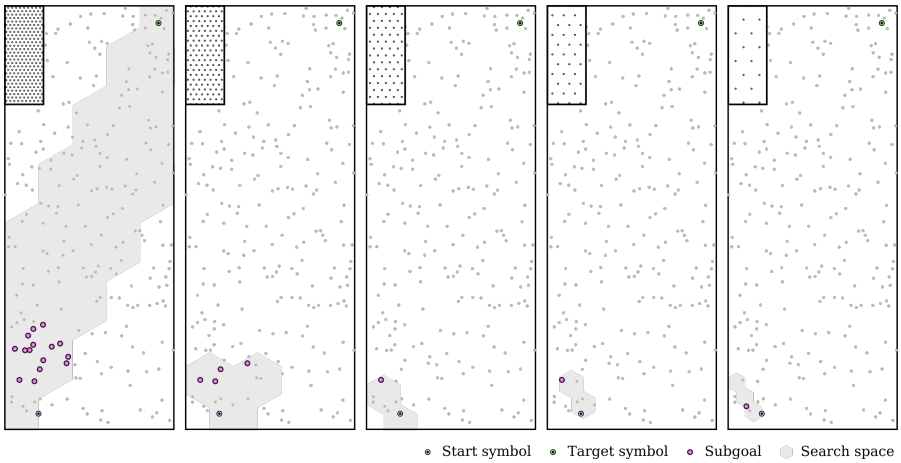

Figure 8: Searching a viable sequence from start (lower left) to target (upper right) using descending mode. The search from start (blue symbol) to target (green symbol) starts on the highest available scale (left panel) and generates subgoals (magenta symbols). The solution is refined using finer scales, which compute only partial solutions to the subgoals from the next higher scale (remaining panels). Distribution of transition encoders is shown in the small inlays in each panel.

an environment of $2 \times 5\,\mathrm{m}^2$, with a minimal distance of 0.05 m between symbols. Then, a transition scale-space consisting of 5 scales with periods $0.2, \sqrt{2} \cdot 0.2, 0.4, \sqrt{2} \cdot 0.4$ and 0.8 m was computed offline.

The initial symbol was determined by finding the closest symbol to coordinate $0.4, 0.1$, and the target by the symbol closest to coordinate $1.8, 4.8$. During search, 3 expansions were computed on each scale before ascending the search to the next scale.

An example of the ascending mode is depicted in Figure 9. Each column shows the expansion on one scale. The first scale searches locally in the area that is indicated by the gray hexagons. Subsequent scales expand the respective search radius, and propagate the search outward until the target is found on the highest scale. For each scale, black dots indicate the symbols that were active at the end of the search on this scale. After finding the target symbol, 100 Monte Carlo trajectory samples were drawn by backtracking from the target symbol to the start symbol, and selecting a random parent during each backtracking step. Note that this results in small trajectory segments and low variability close to the start of the trajectory, but larger segments and higher variability close to the target. In the figure, these samples are illustrated as blue lines on top of the search of the largest scale.

5.5 Shortcut discovery in the Morris water maze experiment

A simulated version of the Morris water maze experiment was used to examine the capability to find shortcuts in spatial information using a transition scale-space. Qualitative results are presented in Figure 10 for three independent simulations. In each simulation, an agent started

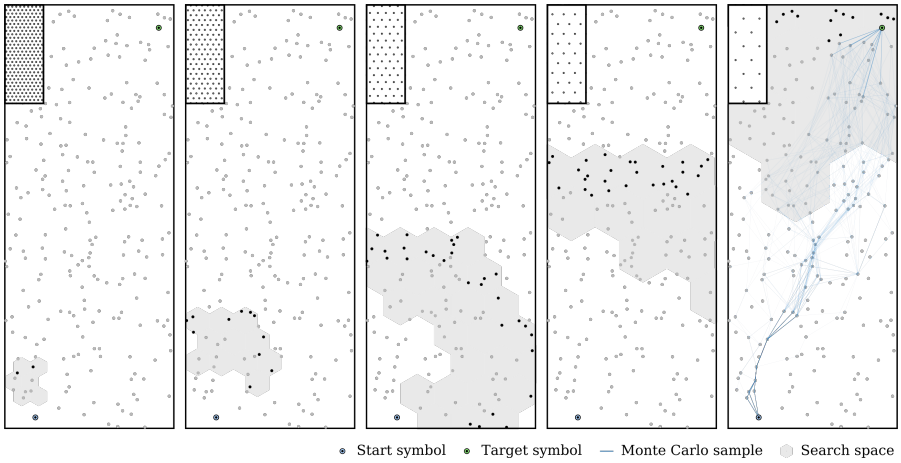

Figure 9: Searching a viable sequence from start (lower left) to target (upper right) using ascending mode. Each scale is expanded at most 3 times (gray area) before ascending to the next scale (panels from left to right), until the highest scale is reached. Black dots indicate the last symbols that were active on this scale before ascending to the next scale or the target was encountered. The rightmost panel includes Monte Carlo samples (blue lines), generated using the backtracking algorithm. Distribution of transition encoders is shown in the small inlays in each panel.

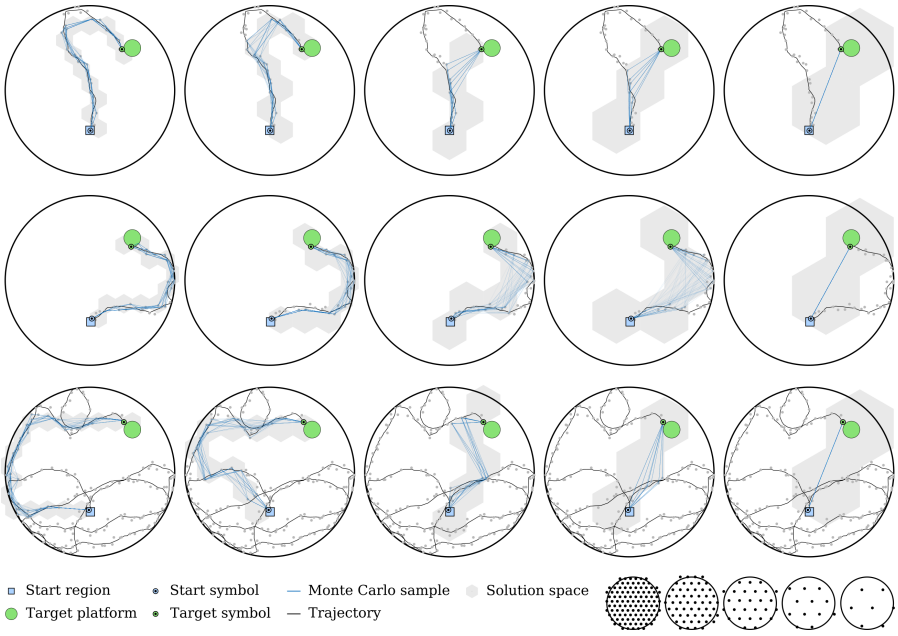

Figure 10: Results of the simulated Morris water maze experiment. Each row represents one experiment, each column using one specific scale to compute the solution space (gray hexagons) and 100 candidate solutions (blue lines). The legend shows the distribution of transition encoders for each scale. See Subsection 5.5 for further details.

from the initial location (blue square) and travelled the environment with a radius of 2 m until the target platform (green circle) was found using movement statistics similar to real rodents. Symbols (or place cells; gray dots) were distributed along the recorded trajectory (black line) as follows, inspired by Growing Neural Gas (GNG)(Fritzke, 1995). As soon as the agent was further away than $m_{\text{dist}} = 0.05\,\text{m}$ from the closest symbol, a new symbol was recruited. To introduce stochastic symbol placement, this novel symbol was associated with a coordinate that was drawn from a unit circle centered at the current position of the agent and normally distributed with a standard deviation of $\frac{m_{\text{dist}}}{20.0}$, where the denominator was chosen arbitrarily. Subsequently, a transition scale-space with 5 scales was computed offline for grid periods of $0.2, \sqrt{2} \cdot 0.2, 0.4, \sqrt{2} \cdot 0.4$ and 0.8 m (distribution of centers shown in the legend of the figure). Specifically, each symbol was associated with the domain of the closest transition encoder, and to the image of the surrounding encoders.

For each scale individually, the solution space was computed according to Algorithm 2. That is, starting from the initial symbol, a sequence of viable transitions and symbols was computed until activation of the target symbol, which was defined as the symbol closest to the target platform. Figure 10 shows the solution space as gray hexagons. For each scale, 100 Monte Carlo samples (blue lines) were computed by backtracking from the target symbol to the initial symbol and selecting a parent symbol randomly during each backtracking step.

The figure shows that, on the smallest scale, reproduced sequences and the solution space remain close to a shortest path on the original trajectory. On higher scales, sample sequences and the solution space exhibit shortcut characteristic, essentially crossing parts of the environment that were not explored previously by the agent.

6 Discussion

This paper introduced scale-space MTS to remedy behaviorally questionable runtimes in model-based algorithms for goal-directed sequence computation. The work assumed that knowledge in the Hippocampus is represented in the form of a topological map, which was previously suggested in Y. Dabaghian et al., 2012 and Yuri Dabaghian et al., 2014. Although the results were resented primarily in the context of spatial navigation due to the nature of GCs and PCs in the Hippocampal formation, the results apply to arbitrary transition systems that operate on topological data. Now, these results will be discussed concerning their plausibility, predictions, and relation to other work.

Specifically, this section is organized as follows. First, the data structure, algorithms, and results of the previous sections will be discussed primarily with respect to their biological plausibility. Then, scale-space MTS will be related to other areas of research, in particular to results from signal processing, mobile robotics, and computer science.

Subsequently, a novel interpretation for Theta and TPP will be presented. Following, the similarities and differences to other transition models of GC will be discussed. Finally, the section closes with future directions.

Note that throughout the section, the terms symbol and neuron may be used interchangeably depending on context.

6.1 Scale-space computations, behavior, and conditions for biological plausibility

Scale-space MTS can explain observations in data from real rodents. For instance, the activity of symbols in an MTS forms a traveling wave during expansion from start to target, depicted in Figure 7. Also the Hippocampal formation shows traveling waves of activity (Lubenov and Siapas, 2009; Patel et al., 2012). However, the simulations showed that an MTS activates all intermediate symbols on the traveling wave until the target is reached. This is unlikely to occur in a biological neural network, as this would suggest a saturation of activity in the network. Rather, it is expected that in a biological network only a sparse activation of neurons in the domain of a transition is sufficient to trigger the transition and to keep the overall activity in the network at an minimal necessary energetic level. Also, it is expected that only a sparse subset of neurons in the image of a transition will become active. This poses the danger that transition neurons miss to activate symbol neurons close to a goal, which is however necessary to terminate the goal-directed search. By over-representing goal locations with a significantly larger number of symbolic neurons than usual, which was observed to be the case in spatial navigation tasks in rodents (Hollup et al., 2001), this issue can be remedied.

A neural network implementation of a scale-space MTS in the form of interval skip lists or segment trees requires only a comparatively simple neural micro-circuit. First, consider a regular skip list. This data structure uses a comparison operation to decide if the process should drop down a layer to find a target. Although this is straightforward to implement in regular programming languages, and could, in principle, also be implemented in neural networks, the necessary operations would require an intricate neural architecture. For instance, such a network would have to learn a partial order to perform the comparison and have an intricate micro-circuit for backtracking to previous elements in the case of overshooting on a scale. Also, a regular skip list only informs about a specific item, which is insufficient for an animal's behavior if it wishes to reproduce an entire sequence. In contrast, interval skip lists appear beneficial on this account. Specifically, learning that a symbol falls into the domain of a spatial transition on a higher scale follows exactly the same rules as lower scales and, thus, generalizes well. That is, symbols are associated to the domain of a transition due to co-activation learning and interval testing therefore also happens due to (co-)activation only. Moreover, learning transitions in multiple scales

can be performed entirely in parallel and independently of each other. Thus, a neural network implementation of a scale-space MTS needs to only differ in the size of the spatial sampling process on each scale. In addition, the extension of Dijkstra's algorithm to operate on sets of symbols is highly suitable for parallelization in a neural network due to simultaneous activity of neurons. Note, however, that an individual symbol not necessarily has to be represented by a singular neuron, but could also be the result of a distributed code, e.g., of an associative memory (Knoblauch, 2005).

The acquired transition information in a scale-space MTS is independent of movement speed or any policy that was used to traverse an environment. However, the knowledge that is available in a scale-space MTS can be used to infer costs for arbitrary sequences. This is due to the constant cost assumption that is fundamental to MTS and the result of optimally bundling transitions. During the expansion process, each additional expansion thereby increases the cost uniformly. Expansions on different scales contribute approximate costs to the estimate, which depends only on the respective scale. Thus, scale-space MTS can provide a sense of cost-to-go to navigating animals that is independent of the reward structure of an environment or their current movement speed.

A scale-space MTS can be used to quickly compute approximate sequences, which is of utmost interest in some behavioral tasks. For instance, consider the Morris water maze experiment, in which a rodent has to establish a route to reach the target platform. Waiting for an accurate computation to finish is not an option for a rodent, given the perilous situation in which it is during this experiment. Rather, the animal requires an algorithm that quickly yields an estimate of the solution so that it can start moving, and refine the trajectory online. The qualitative results shown in Figure 10 show that a scale-space MTS provides such an algorithm. Using a large scale quickly narrows the search space, or is even able to present a direction directly towards the goal. Note that a large scale comes at the cost of an increased variance of the solution, indicated by the large gray areas in the figures. Therefore, it is expected that rodents operate not only on the highest available scale, but either on all scales in parallel, or in ascending mode. The benefit of the latter is that it gives a detailed solution of a goal-directed sequence for the immediate next steps, but requires only limited computational time. Due to the low computational cost, the approximations can be refined online, or quickly recomputed in the case of changing environmental conditions.

Sequence selection was computed using classical backtracking in the results presented in Section 5. In this form, the algorithm is not biologically plausible. The following will sketch an implementation to address this issue. First, recall that MTS require that for any transition to have an effect, both the transition as well as the parent symbol from

which the transition originates must be active or logically true. Thus, parent information is locally available after a symbol becomes active, which could lead to a short term change of the synaptic strength between the two symbols. In addition, assume that monitoring the target symbol means that the synaptic strength between the target symbol and any of its parent symbols gets increased for a brief temporal duration. Now, reconsider the constant cost assumption of MTS. Essentially, this means that the cost to reach neighboring symbols is constant on one scale. This can be interpreted as a temporal delay or a certain threshold to activate expanded symbols. Increasing the strength of a connection between symbols should therefore lead to a decreased temporal delay before an expanded symbol activates. After only a few iterations, the optimal solution from start to target will therefore be the one that is the shortest in execution time. Note that this mechanism could explain why rats appear to preplay multiple similar trajectories during homing tasks, observed by Pfeiffer and Foster, 2013. Moreover, iteratively repeating this process should shift the activity of place cells on a solution trajectory closer in time to the activity of a starting symbol. Finally, this idea of a biologically plausible backtracking can be improved with fast local inhibitory circuits to quickly discard non-optimal solutions. That is, a symbol that activates sufficiently fast may be able to suppress competing representations before they get expanded. Note that this could also present a mechanism by which rewards modulate activity of symbols and transitions. Implementation of these ideas will be left for future work, though.

6.2 Scale-spaces, data structures, and graph algorithms

The constructive procedure presented in Section 4 generates a data structure that contains transitions in multiple resolutions. Specifically, it forms a scale-space in the form of a Gaussian pyramid. This type of data structure is well known in the computer vision and signal processing communities (T. Lindeberg, 1994; Tony Lindeberg, 2010; Witkin, 1983). Among other application areas, Gaussian pyramids were used to describe biologically plausible retinal and visuocortical receptive fields (S. Behnke and Rojas, 1998; Georgeson et al., 2007; Young, 1987). In addition, it was shown previously that a scale increment of $\sqrt{2}$ is optimal for normally distributed inputs in vision tasks (Tony Lindeberg, 2010). In image processing, application of a Gaussian pyramid corresponds to consecutively smoothing an image, i.e. an input gets low-pass filtered by which fine-scale information is removed. Consequently, this allows to detect features in a scale-invariant way, which is commonly used in classical computer vision algorithms (Tony Lindeberg, 2015; Lowe, 2004).

The same intuitive idea is certainly true for spatial navigation. To plan sequences to remote locations, it appears behaviorally unnecessary to factor fine details close to the target into the solution. Rather, it

seems more important to quickly establish an approximate solution, which can be refined during execution. In other cases, fine details may not be relevant at all, and approximate solutions entirely sufficient. Gaussian pyramids provide a principled solution to compute these approximations, as demonstrated in the examples presented in this paper.

The computational benefits of spatial approximations with and clustering into multiple scales of representation is also well known in other areas. For instance, the computer graphics community has widely adopted data structures that are based on or extensions of quad-trees (Finkel and Bentley, 1974), or higher dimensional variants such as octrees (Orenstein, 1982), to achieve real-time graphics. Also the robotics and path planning communities have adopted multi-scale data structures (Sven Behnke, 2004; Hwang et al., 2003; Kambhampati and Davis, 1986), which are often combined with (extensions to) classical algorithms such as Dijkstra's algorithm (Dijkstra, 1959).

The method presented in this paper is related to these classical graph algorithms, and in particular A* (Hart et al., 1968). A* generates optimal solutions during graph searches, but its runtime crucially depends on a suitable heuristic. The basic idea of A* is the same as in Dijkstra's algorithm, meaning that the algorithm sequentially expands nodes and tests if they are on a path to the target. In contrast to Dijkstra's algorithm, A* uses a heuristic that guides the expansion of nodes and thereby limits the search space significantly. Still, A* requires that the heuristic is admissible, which means that the heuristic always has to underestimate the real distance to the target. The scale-space approach presented in this paper adheres to this condition, and is thus a suitable heuristic for A*.

6.3 Spatial pooling, temporal buffering, and Theta Phase Precession

The transition scale-space was computed offline for the experiments presented in Section 5. An animal, however, has to learn the data structure in a bottom-up manner and, importantly, in a way that is suitable for synaptic plasticity.

Imagine a scale-space MTS that is implemented in a spiking neural network and that is governed by Spike-Timing Dependent Plasticity (STDP). To learn transitions from one set of symbols to another set of symbols, all neurons that represent symbols in the domain of the transition have to spike before the transition neuron. In contrast, symbols in the image of the transition have to spike only after the transition neuron (see panel A of Figure 11). If, however, symbols were only active when they were perceived, then they would not co-activate with the transition neuron's integration area. Thus, there must be a mechanism that allows transition neurons of higher scales to associate with symbols that correspond to past and future locations.

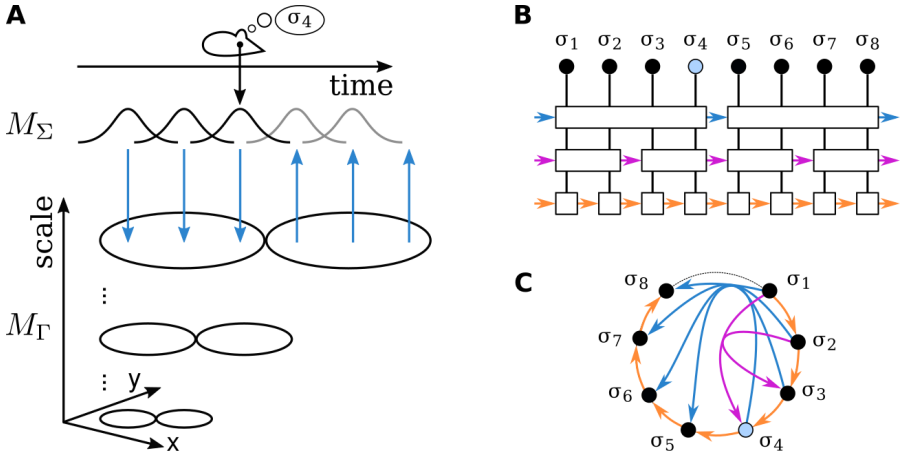

Figure 11: Scale-space learning, temporal buffering, and Theta Phase Precession. (A) Transition encoders of M_Γ learn transitions in both the spatial input space and the symbolic space of M_Σ simultaneously. To form higher scales, access to temporally adjacent symbols relative to the current position (σ_4) is required. (B) One dimensional example of the formation of the scale-space structure on discrete elements. The transition encoder on the highest scale requires access to symbols $\sigma_1, \ldots, \sigma_4$ to learn its domain, and to $\sigma_5, dots, \sigma_8$ to learn its image (only rightwards direction of the transition shown here). (C) For association learning, the symbols need to co-activate with the transition encoder in a way that is suitable for synaptic plasticity. A change of perspective reveals that this can be achieved by a temporal ring buffer that oscillates relative to the current symbol. For instance, symbols $\sigma_5, \ldots, \sigma_8$, which are spatially ahead of σ_4, are temporally ordered with respect to their spatial occurrence after σ_4.

Theta oscillations appear to provide a suitable solution to this problem, and TPP is an observable effect of it. Namely, Theta temporally buffers symbols that represent past locations, and pre-fetches symbols that represent future locations on the current trajectory of the animal. This can be achieved by iterating the transition system on the smallest scale, possibly with a dependence on the heading direction of the animal to retrieve only locations which are on a linear future track. Consider an animal that is at a location which is mentally represented by symbol σ_4 and living in a discretized one-dimensional world, illustrated in panel B of Figure 11. To learn a large scale transition, for instance on the highest scale of the figure, a neuron that represents such a transition necessarily needs all spikes from $\sigma_1, \ldots, \sigma_4$ to arrive before it spikes, and spikes from $\sigma_5, \ldots, \sigma_8$ to arrive only after. In other words, this implies a temporal order of the elements relative to their spatial occurrence. Moreover, this buffer needs to capture those symbols that are relevant to learn the highest scale, after which it can repeat itself. Changing the illustration reveals that this can be performed using a temporal ring buffer that oscillates relative to the current symbol (panel C of Figure 11). Conclusively, this allows to interpret Theta as the observable effect of this temporal ring buffer.

Theta was previously described as a propagating wave in the Hippocampus (Lubenov and Siapas, 2009). This supports the speculation that Theta is the observable effect that is generated by expanding nodes in a topographical map to learn large-scale transitions. Consider the results of the wave-propagation experiment depicted in Figure 7. Not only is the expansion a traveling wave towards a target location. Also, the number of active symbols increases during each expansion step. In combination with the temporal ring buffer that effectively limits the expansion, but also has a singular point in the representation of the currently active location, this could lead to an explanation for the oscillating change of the local field potential in the Hippocampus.

In addition to a temporal ring buffer, the brain may rely on other mechanisms to support the formation of large scale transition knowledge, for instance Sharp Waves and Ripples (SPW-R) (see review in Colgin, 2016 for an overview of rhythms related to Theta). During SPW-R, actually travelled trajectories are replayed in a temporally compressed manner, which was found to support the consolidation of spatial memories (Jadhav et al., 2012). It is exactly this form of temporal compression that is necessary to form large scale transition knowledge that would otherwise not fit into the temporal integration window of STDP.

As of now, this interpretation of Theta and TPP leaves several question unanswered. For instance, it is unclear how the discontinuity between predicting and prefetching symbols in the temporal ring buffer is achieved. Also, possible interactions between memories M_Γ and M_Π are not yet unexplored. It is likely, however, that these memories interact in a way that supports spatio-temporal coherence.

6.4 Relationship to other vector-based navigation and successor representations

The transition scale-space is related to several other models of GCs, and in particular to models of vector-based navigation in the rodent brain. In these models, multiple scales of GCs are commonly used to encode a single spatial location, or a vector to a location. For instance, the HiLAM model plans trajectories by first computing approximate locations using the highest scale of its representation, and narrows down on the exact position using finer scales. However, such representations will yield multiple feasible vectors to different locations given a suitably large environment. To circumvent this problem, large scale representations are mandatory in vector-based navigation models. In contrast, scale-space MTS does not suffer from such ambiguities because it only stores local transitional information and not spatial locations. Moreover, large scale representations are not necessary for accurate and decisive navigational solutions, but rather to achieve behaviorally relevant computational runtimes.

The scale-space MTS model of GCs is also related to other transition models of GCs, namely PCA or SR based models (Dordek et al., 2016; Momennejad et al., 2016; Stachenfeld et al., 2016). These models compute grid like responses based on PC inputs and represent correlate information between GC and PC activity in multiple scales. For instance, higher scales of SR models represent transitions to remote locations and learn this information exclusively on PC afferents. However, GCs of the scale-space MTS model learn their representation on a suitable input space and link to PCs only due to co-activation learning. Thereby, they relay structural information from the input space to higher cortical representations, which can then be used independent of the input space during retrieval (see Waniek, 2018 for a longer discussion thereof). In addition, SR models propose that grid cells learn to represent the weighted sum of rewards of distant future states. That is, these models implicitly depend on a certain policy by which the animal navigates the environment and require that an environment and its intermediate states are sampled sufficiently often for the representation to converge. Moreover, the animal needs to learn a distinct SR representations for each specific goal. Given recent evidence that PCs are insensitive to reward information (Duvelle et al., 2019), it remains to determine if GCs support reward storage, or in which way reward information could be available to support learning of an SR. In contrast, scale-space MTS is independent of any policy or reward and can be recomputed easily for arbitrary targets. Also, MTS requires only limited information about traversed states and does not require dense coverage of the input, as demonstrated in the Morris water maze experiment presented in Section 5. Nevertheless, SR models have the benefit that they can compute trajectories to target locations without any additional reward mechanism. Possible future work therefore includes to combine the benefits of both models.

6.5 Future directions

The results presented and discussed in this paper lead to several follow-up questions from the perspective of computational neuroscience. For instance, (i) how does an implementation of a scale-space MTS look like and behave in biologically plausible, spiking neural networks, and how does this implementation relate to TPP; (ii) can MTS be extended to conjunctive representations; (iii) is this fundamental data structure also implemented in other cortical areas that require sequence learning and planning, such as language understanding and generation, or motion prediction in the visual cortex; (iv) is it possible to extract probabilistic models of the suggested mechanism; or (v) can transitions in the form of grid codes and spatial representations in the form of place codes be learned bottom-up from navigational tasks without defining the data structures beforehand, to name just a few.

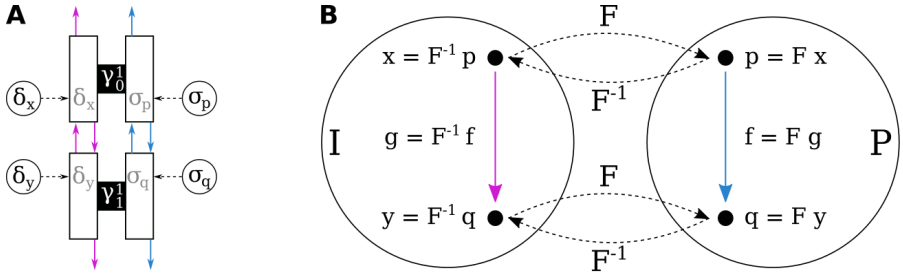

Figure 12: Transitions, GCs, and category theory. (A) Visualization of GCs according to scale-space MTS that learn transitions from symbols δ_x to δ_y, while also learning transitions from σ_p to σ_q. (B) Commutative diagram of category theory, in which a morphism g maps objects x to y in a category I, a morphism f maps objects p to q in a category P. The morphisms and objects are linked by a functor F that preserves structure of the categories.

Some of these questions were – at least partially – already addressed by others. While (i) and (ii) will be left for future work, evidence shows that grid codes or grid-like representations exist in areas that are involved in other computations than just spatial navigation (Nau et al., 2018), and thus support the claim for (iii). Significant contributions into the direction of (iv) were presented by Stachenfeld et al., 2016, Momennejad et al., 2016, and Dordek et al., 2016, who argued that GCs encode PCA or SRs and provide probabilistic formulations for their approaches. More recently, also Whittington et al., 2018 argued along the same lines and presented an elegant mathematical description of their model. It thus appears natural to extend scale-space MTS to a purely probabilistic formulation for all involved steps. Nevertheless, it remains for future work to understand if either SRs, MTS, or a combination of the two yield more insight in and understanding of GCs and PCs, and consistency with recordings from real animals. Considering (v), Cueva and X.-X. Wei, 2018 and, independently, Banino et al., 2018 observed that training deep neural networks on spatial navigation tasks leads to the formation of place and grid representations. The latter in particular argue that their system learns vector-based navigation and spatial codes. However, a closer examination of their model architecture reveals that the cells which exhibit grid codes learn transitions for downstream spatial representations. Hence, grid representations appear to be, in fact, an optimality result when learning from navigational tasks in recurrent networks.

Extending the transition scale-space with a reward propagation mechanism may solve current issues in RL. For instance, recent progress in the domain of multi-agent RL has been shown to depend crucially on additional temporal scales to transport rewards (Jaderberg et al., 2018). The reason is that rewards are typically sparse in real-world scenarios, which poses an issue for most RL approaches and is therefore an active field of research. In their model, Jaderberg et al., 2018 used an additional temporal scale to solve this problem. In contrast to a solution

that is specifically tailored for a distinct scenario, the transition scale-space, augmented with a reward transportation mechanism, potentially solves this issue in a general manner.

Finally, a novel direction is revealed by the observation that transition systems have a relationship to category theory. Although category theory is exceptionally abstract (Adamek et al., 2009), this perspective might open up a new understanding of general cortical computations using fundamental mathematical principles. Changing the terms that were used throughout this paper makes the link to category theory more pronounced. Let any transition between two symbols of the same alphabet be a *morphism* that operates on some objects. For instance, spatial transitions are morphisms $g : x \rightarrow y$ that operate on objects x, y of the category I of spatial symbols. Learning spatial transitions, part of the proposed dendritic computation of GCs, is thus learning of morphisms in the category I. In MTS however, GCs simultaneously also learn transitions in the representation that is created by place cells, i.e., morphisms $f : p \rightarrow q$ that operate on the category of place symbols P. Observe that according to scale-space MTS – but also in the context of SRs – GCs learn to preserve the structure of the input space and convey this knowledge to place cell representations. Consequently, GCs can be interpreted as *functors* F that transport objects $x \in I$ to objects $Fx \in P$ and morphisms from the category of spatial symbols I to morphisms in the category of place cells P. Namely, they transport transitions between symbols from one category to the other, that is $F\,g : F\,x \rightarrow F\,y$, while preserving structure. Also, for every functor F, there exists an inverse functor F^{-1}, i.e. $F^{-1}\,f : F^{-1}\,p \rightarrow F^{-1}\,q$. Note that these observations apply across different scales of transition representations, regardless of the size of the grid field, and thus expose a fundamental cortical operation. Figure 12 illustrates the relationship between MTS and category theory. During spatial navigation, the existence of the inverse functor allows to perform inference on the sensory inputs $\delta_i in \Delta$ given a sequence symbols $\sigma_k \in \Sigma$, demonstrated by computing the solution space in the examples of Section 5. Certainly, also other cortical areas have to learn relationships between objects of different categories while preserving structure, for instance to link the visual representation of an "apple" with the word "apple". To conclude this argument, it appears reasonable to investigate if cortical computations can be understood and expressed compactly in terms of category theory.

7 Conclusion and outlook

In this paper, transition encoding was examined from the perspective of behavioral requirements for path planning. An example showed that a single scale of transition encoders is insufficient to provide run-times that lie in a meaningful range when neuronal axonal delays are respected. However, transition encoding allows interpretation in form of linked lists of data, for which asymptotically optimal search strategies

are well known from computer science. In particular, interval skip lists introduce hierarchies to accelerate searches exponentially. Under consideration of neural encodings that follow a Gaussian distribution, it was presented that the optimal scale increment for such techniques is $\sqrt{2}$. The resulting hierarchical representation to accelerate path planning by transition look-ahead forms a scale-space that provides approximate path planning solutions. The data structure and algorithms were evaluated using simulations. Given that scale-spaces are well known from other cortical regions, primarily the visual cortex, this allowed to discuss the results in terms of a general principle that underlies cortical processing. Furthermore, the transition scale-space was related to techniques for multi-resolution path planning that are well-established in the robotics community. Conclusively, the discretized EC could be explained as an effective, yet general purpose, data structure that achieves optimal run-time behavior during goal-directed sequence planning.

Future work will include a biologically plausible model for the self-organization of the dendritic tree of GCs and, thus, the formation of hexagonal grid fields. Moreover, the transition scale-space will be extended with reward propagation mechanisms and a probabilistic model of computation.

Acknowledgments

The author would like to thank Christoph Richter and Jörg Conradt for discussions and feedback on the original idea of multi-transition encoding. Also, thanks to Christoph Schied for invaluable discussions on acceleration data structures and algorithms. Finally, thanks to Maike Krause for significant contributions to the structure of the manuscript.

The work presented in this paper was conceived primarily while the author was with the Neuroscientific System Theory Group (NST), Technische Universität München, Arcisstraße 21, 80333 München, Germany, and was partially funded by EU FET project GRIDMAP 600725. The author declares no competing financial interest.

References

Adamek, J., Herrlich, H., & Strecker, G. E. (2009). *Abstract and concrete categories: The joy of cats (dover books on mathematics)*. Dover Publications. Retrieved from https://www.amazon.com/Abstract-Concrete-Categories-Dover-Mathematics/dp/0486469344?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0486469344

Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., ... Kumaran, D. (2018). Vector-based navigation using grid-like

representations in artificial agents. *Nature*, *557*(7705), 429–433. doi:10.1038/s41586-018-0102-6

Behnke, S. [S.], & Rojas, R. (1998). Neural abstraction pyramid: A hierarchical image understanding architecture. In *1998 ieee international joint conference on neural networks proceedings. ieee world congress on computational intelligence (cat. no.98ch36227)* (Vol. 2, 820–825 vol.2). doi:10.1109/IJCNN.1998.685873

Behnke, S. [Sven]. (2004). Local multiresolution path planning. In D. Polani, B. Browning, A. Bonarini, & K. Yoshida (Eds.), *Robocup 2003: Robot soccer world cup vii* (pp. 332–343). Berlin, Heidelberg: Springer Berlin Heidelberg.

Bender, M. A., Farach-Colton, M., Johnson, R., Mauras, S., Mayer, T., Phillips, C. A., & Xu, H. (2017). Write-optimized skip lists. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems - PODS '17*. doi:10.1145/3034786.3056117

Bonnevie, T., Dunn, B., Fyhn, M., Hafting, T., Derdikman, D., Kubie, J. L., . . . Moser, M.-B. (2013). Grid cells require excitatory drive from the hippocampus. *Nat Neurosci*, *16*(3), 309–317. doi:10.1038/nn.3311

Burak, Y., & Fiete, I. R. (2009). Accurate path integration in continuous attractor network models of grid cells. *PLOS Computational Biology*, *5*(2), 1–16. doi:10.1371/journal.pcbi.1000291

Bush, D., Barry, C., & Burgess, N. (2014). What do grid cells contribute to place cell firing? *Trends Neurosci. 37*(3), 136–145.

Butts, D. A., & Goldman, M. S. (2006). Tuning curves, neuronal variability, and sensory coding. *PLoS Biol. 4*(4), e92.

Colgin, L. L. (2016). Rhythms of the hippocampal network. *Nature Reviews Neuroscience*, *17*(4), 239–249. doi:10.1038/nrn.2016.21

Couey, J. J., Witoelar, A., Zhang, S. J., Zheng, K., Ye, J., Dunn, B., . . . Witter, M. P. (2013). Recurrent inhibitory circuitry as a mechanism for grid formation. *Nat. Neurosci. 16*(3), 318–324.

Cover, T. M., & Thomas, J. A. (2006). *Elements of information theory (wiley series in telecommunications and signal processing)*. Wiley-Interscience.

Cueva, C. J., & Wei, X.-X. (2018). Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. In *International conference on learning representations*. Retrieved from https://openreview.net/forum?id=B17JTOe0-

Cuperlier, N., Laroque, P., Gaussier, P., & Quoy, M. (2004). Planning and navigation strategies using transition cells and neural fields. In *Proc. of asc/iasted 2004 (artificial intelligence and soft computing / international association of science and technology for development)*, Marbella. Retrieved from http://publi-etis.ensea.fr/2004/CLGQ04

Cuperlier, N., Quoy, M., Giovannangeli, C., Gaussier, P., & Laroque, P. (2006). Transition cells for navigation and planning in an unknown

environment. In S. Nolfi, G. Baldassarre, R. Calabretta, J. C. T. Hallam, D. Marocco, J.-A. Meyer, . . . D. Parisi (Eds.), *From animals to animats 9: 9th international conference on simulation of adaptive behavior, sab 2006, rome, italy, september 25-29, 2006. proceedings* (pp. 286–297). doi:10.1007/11840541_24

Dabaghian, Y. [Y.], Memoli, F., Frank, L., & Carlsson, G. (2012). A topological paradigm for hippocampal spatial map formation using persistent homology. *PLoS Comput. Biol. 8*(8), e1002581.

Dabaghian, Y. [Yuri], Brandt, V. L., & Frank, L. M. (2014). Reconceiving the hippocampal map as a topological template. *eLife*, *3*, e03476. doi:10.7554/eLife.03476

Davoudi, H., & Foster, D. J. (2019). Acute silencing of hippocampal CA3 reveals a dominant role in place field responses. *Nature Neuroscience*. doi:10.1038/s41593-018-0321-z

Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, *5*(4), 613–624. doi:10.1162/neco.1993.5.4.613

de Berg, M., van Kreveld, M., Overmars, M., & Schwarzkopf, O. (1997). *Computational geometry*. doi:10.1007/978-3-662-03427-9

Dean, B. C., & Jones, Z. H. (2007). Exploring the duality between skip lists and binary search trees. In *Proceedings of the 45th annual southeast regional conference* (pp. 395–399). ACM-SE 45. doi:10.1145/1233341.1233413

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271. doi:10.1007/bf01386390

Dordek, Y., Soudry, D., Meir, R., & Derdikman, D. (2016). Extracting grid cell characteristics from place cell inputs using non-negative principal component analysis. *eLife*, *5*. doi:10.7554/elife.10094

Duvelle, É., Grieves, R. M., Hok, V., Poucet, B., Arleo, A., Jeffery, K., & Save, E. (2019). Insensitivity of place cells to the value of spatial goals in a two-choice flexible navigation task. *The Journal of Neuroscience*, 1578–18. doi:10.1523/jneurosci.1578-18.2018

Eppstein, D., Goodrich, M. T., & Sun, J. Z. (2005). The skip quadtree. In *Proceedings of the twenty-first annual symposium on computational geometry - SCG '05*. doi:10.1145/1064092.1064138

Erdem, U. M., Milford, M. J., & Hasselmo, M. E. (2015). A hierarchical model of goal directed navigation selects trajectories in a visual environment. *Neurobiology of Learning and Memory*, *117*, 109–121. Memory and decision making. doi:http://dx.doi.org/10.1016/j.nlm.2014.07.003

Felleman, D. J., & Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cereb. Cortex*, *1*(1), 1–47.

Finkel, R. A., & Bentley, J. L. (1974). Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, *4*(1), 1–9. doi:10.1007/BF00288933

Fritzke, B. (1995). A growing neural gas network learns topologies. In *Advances in neural information processing systems 7* (pp. 625–632). MIT Press.

Fuhs, M. C., & Touretzky, D. S. (2006). A spin glass model of path integration in rat medial entorhinal cortex. *J. Neurosci. 26*(16), 4266–4276.

Fyhn, M., Solstad, T., & Hafting, T. (2008). Entorhinal grid cells and the neural basis of navigation. In *Hippocampal place fields* (pp. 237–252). doi:10.1093/acprof:oso/9780195323245.003.0018

Georgeson, M. A., May, K. A., Freeman, T. C. A., & Hesse, G. S. (2007). From filters to features: Scale–space analysis of edge and blur coding in human vision. *Journal of Vision*, *7*(13), 7. doi:10.1167/7.13.7

Giocomo, L. M., Moser, M.-B., & Moser, E. I. (2011). Computational models of grid cells. *Neuron*, *71*(4), 589–603. doi:http://dx.doi.org/10.1016/j.neuron.2011.07.023

Gustafson, N. J., & Daw, N. D. (2011). Grid cells, place cells, and geodesic generalization for spatial reinforcement learning. *PLoS Computational Biology*, *7*(10), e1002235. doi:10.1371/journal.pcbi.1002235

Hafting, T., Fyhn, M., Molden, S., Moser, M.-B., & Moser, E. I. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature*, *436*(7052), 801–806. doi:10.1038/nature03721

Hanson, E. N. (1991). The interval skip list: A data structure for finding all intervals that overlap a point. In F. Dehne, J. Sack, & N. Santoro (Eds.), *Algorithms and data structures* (pp. 153–164). doi:10.1007/bfb0028258

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, *SSC-4*(2), 100–107.

Hirel, J., Gaussier, P., Quoy, M., & Banquet, J.-P. (2010). Why and how hippocampal transition cells can be used in reinforcement learning. In S. Doncieux, B. Girard, A. Guillot, J. Hallam, J.-A. Meyer, & J.-B. Mouret (Eds.), *From animals to animats 11: 11th international conference on simulation of adaptive behavior, sab 2010, paris - clos lucé, france, august 25-28, 2010. proceedings* (pp. 359–369). doi:10.1007/978-3-642-15193-4_34

Hollup, S. A., Molden, S., Donnett, J. G., Moser, M.-B., & Moser, E. I. (2001). Accumulation of hippocampal place fields at the goal location in an annular watermaze task. *The Journal of Neuroscience*, *21*(5), 1635–1644. doi:10.1523/jneurosci.21-05-01635.2001

Hwang, J. Y., Kim, J. S., Lim, S. S., & Park, K. H. (2003). A fast path planning by path graph optimization. *IEEE Trans. Systems, Man, and Cybernetics, Part A*, *33*(1), 121–129. doi:10.1109/TSMCA.2003.812599

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castaneda, A. G., . . . Graepel, T. (2018). Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. eprint: arXiv:1807.01281

Jadhav, S. P., Kemere, C., German, P. W., & Frank, L. M. (2012). Awake hippocampal sharp-wave ripples support spatial memory. *Science*, *336*(6087), 1454–1458. doi:10.1126/science.1217230

Jazayeri, M., & Movshon, J. A. (2006). Optimal representation of sensory information by neural populations. *Nat Neurosci*, *9*(5), 690–696. doi:10.1038/nn1691

Jeewajee, A., Barry, C., Douchamps, V., Manson, D., Lever, C., & Burgess, N. (2013). Theta phase precession of grid and place cell firing in open environments. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, *369*(1635). doi:10.1098/rstb.2012.0532

Kambhampati, S., & Davis, L. S. (1986). Multiresolution path planning for mobile robots. *IEEE J. Robotics and Automation*, *2*(3), 135–145. doi:10.1109/JRA.1986.1087051

Knoblauch, A. (2005). Neural associative memory for brain modeling and information retrieval. *Information Processing Letters*, *95*(6), 537–544. Applications of Spiking Neural Networks. doi:http://dx.doi.org/10.1016/j.ipl.2005.05.021

Knuth, D. E. (1998). *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc.

Kropff, E., Carmichael, J. E., Moser, M.-B., & Moser, E. I. (2015). Speed cells in the medial entorhinal cortex. *Nature*, *523*(7561), 419–424. Article. Retrieved from http://dx.doi.org/10.1038/nature14622

Krupic, J., Bauza, M., Burton, S., Lever, C., & O'Keefe, J. (2014). How environment geometry affects grid cell symmetry and what we can learn from it. *Philos. Trans. R. Soc. Lond., B, Biol. Sci. 369*(1635), 20130188.

Langston, R. F., Ainge, J. A., Couey, J. J., Canto, C. B., Bjerknes, T. L., Witter, M. P., . . . Moser, M. B. (2010). Development of the spatial representation system in the rat. *Science*, *328*(5985), 1576–1580.

Lindeberg, T. [T.]. (1994). Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of Applied Statistics*, *21*, 225–270.

Lindeberg, T. [Tony]. (2010). *Scale-space theory in computer vision (the springer international series in engineering and computer science)*. Springer.

Lindeberg, T. [Tony]. (2015). Image matching using generalized scale-space interest points. *Journal of Mathematical Imaging and Vision*, *52*(1), 3–36. doi:10.1007/s10851-014-0541-0

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, *60*(2), 91–110. doi:10.1023/B:VISI.0000029664.99615.94

Lu, Y., Huo, X., Arslan, O., & Tsiotras, P. (2011). Incremental multi-scale search algorithm for dynamic path planning with low worst-case complexity. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, *41*(6), 1556–1570. doi:10.1109/TSMCB.2011.2157493

Lubenov, E. V., & Siapas, A. G. (2009). Hippocampal theta oscillations are travelling waves. *Nature*, *459*(7246), 534–539. doi:10.1038/nature08010

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. New York, NY, USA: Henry Holt and Co., Inc.

Mathis, A., Herz, A. V. M., & Stemmler, M. (2012). Optimal population codes for space: Grid cells outperform place cells. *Neural Computation*, *24*(9), 2280–2317. doi:10.1162/NECO_a_00319

McNamee, D., Wolpert, D. M., & Lengyel, M. (2016). Efficient state-space modularization for planning: Theory, behavioral and neural signatures. In *Advances in neural information processing systems* (pp. 4511–4519).

McNaughton, B. L., Battaglia, F. P., Jensen, O., Moser, E. I., & Moser, M.-B. (2006). Path integration and the neural basis of the 'cognitive map'. *Nat Rev Neurosci*, *7*(8), 663–678. doi:10.1038/nrn1932

Momennejad, I., Russek, E. M., Cheong, J. H., Botvinick, M. M., Daw, N., & Gershman, S. J. (2016). The successor representation in human reinforcement learning. doi:10.1101/083824

Morris, R. G., Garrud, P., Rawlins, J. N., & O'Keefe, J. (1982). Place navigation impaired in rats with hippocampal lesions. *Nature*, *297*(5868), 681–683.

Moser, E. I. [E. I.], & Moser, M. B. (2008). A metric for space. *Hippocampus*, *18*(12), 1142–1156.

Moser, E. I. [Edvard I.], Roudi, Y., Witter, M. P., Kentros, C., Bonhoeffer, T., & Moser, M.-B. (2014). Grid cells and cortical representation. *Nat Rev Neurosci*, *15*(7), 466–481. Review. Retrieved from http://dx.doi.org/10.1038/nrn3766

Mulas, M., Waniek, N., & Conradt, J. (2016). Hebbian plasticity realigns grid cell activity with external sensory cues in continuous attractor models. *Front Comput Neurosci*, *10*, 13. 26924979[pmid]. doi:10.3389/fncom.2016.00013

Muller, R. U., & Kubie, J. L. (1987). The effects of changes in the environment on the spatial firing of hippocampal complex-spike cells. *J. Neurosci. 7*(7), 1951–1968.

Nau, M., Schröder, T. N., Bellmund, J. L. S., & Doeller, C. F. (2018). Hexadirectional coding of visual space in human entorhinal cortex. *Nature Neuroscience*, *21*(2), 188–190. doi:10.1038/s41593-017-0050-8

O'Keefe, J. (1979). A review of the hippocampal place cells. *Prog. Neurobiol. 13*(4), 419–439.

Orenstein, J. A. (1982). Multidimensional tries used for associative searching. *Information Processing Letters*, *14*(4), 150–157. doi:http://dx.doi.org/10.1016/0020-0190(82)90027-8

Pai, D. K., & Reissell, L. (1995). Multiresolution rough terrain motion planning. In *Proceedings of IEEE/RSJ international conference on intelligent robots and systems, IROS 1995, august 5 - 9, 1995, pittsburgh, pa, USA* (pp. 39–44). doi:10.1109/IROS.1995.526136

Patel, J., Fujisawa, S., Berényi, A., Royer, S., & Buzsáki, G. (2012). Traveling theta waves along the entire septotemporal axis of the hippocampus. *Neuron*, *75*(3), 410–417. doi:10.1016/j.neuron.2012.07.015

Petersen, D. P., & Middleton, D. (1962). Sampling and reconstruction of wave-number-limited functions in n-dimensional euclidean spaces. *Information and Control*, *5*(4), 279–323. doi:http://dx.doi.org/10.1016/S0019-9958(62)90633-2

Pfeiffer, B. E., & Foster, D. J. (2013). Hippocampal place-cell sequences depict future paths to remembered goals. *Nature*, *497*(7447), 74–79.

Pilarski, S., & Pokora, O. (2015). On the cramér–rao bound applicability and the role of fisher information in computational neuroscience. *Biosystems*, *136*, 11–22. Selected papers presented at the Eleventh International Workshop on Neural Coding, Versailles, France, 2014. doi:https://doi.org/10.1016/j.biosystems.2015.07.009

Pugh, W. (1990). Skip lists: A probabilistic alternative to balanced trees. *Commun. ACM*, *33*(6), 668–676. doi:10.1145/78973.78977

Ranck, J. B. (1984). Head-direction cells in the deep cell layers of dorsal presubiculum in freely moving rats. *Society for Neuroscience Abstracts*, *10*.

Rowland, D. C., Roudi, Y., Moser, M.-B., & Moser, E. I. (2016). Ten years of grid cells. *Annual Review of Neuroscience*, *39*(1), 19–40. doi:10.1146/annurev-neuro-070815-013824

Sargolini, F., Fyhn, M., Hafting, T., McNaughton, B. L., Witter, M. P., Moser, M. B., & Moser, E. I. (2006). Conjunctive representation of position, direction, and velocity in entorhinal cortex. *Science*, *312*(5774), 758–762.

Savelli, F., Yoganarasimha, D., & Knierim, J. J. (2008). Influence of boundary removal on the spatial representations of the medial entorhinal cortex. *Hippocampus*, *18*(12), 1270–1282.

Shipston-Sharman, O., Solanka, L., & Nolan, M. F. (2016). Continuous attractor network models of grid cell firing based on excitatory-inhibitory interactions. *J. Physiol. (Lond.) 594*(22), 6547–6557.

Solstad, T., Moser, E. I., & Einevoll, G. T. (2006). From grid cells to place cells: A mathematical model. *Hippocampus*, *16*(12), 1026–1031.

Stachenfeld, K. L., Botvinick, M. M., & Gershman, S. J. (2016). The hippocampus as a predictive map. doi:10.1101/097170

Stemmler, M., Mathis, A., & Herz, A. V. M. (2015). Connecting multiple spatial scales to decode the population activity of grid cells. *Science Advances*, *1*(11). doi:10.1126/science.1500816

Stensola, H., Stensola, T., Solstad, T., Froland, K., Moser, M.-B., & Moser, E. I. (2012). The entorhinal grid map is discretized. *Nature*, *492*(7427), 72–78. doi:10.1038/nature11649

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction (adaptive computation and machine learning series)*. A Bradford Book.

Waniek, N. (2018). Hexagonal grid fields optimally encode transitions in spatiotemporal sequences. *Neural Computation*, *30*(10), 2691–2725. doi:10.1162/neco_a_01122

Wei, X.-x., Prentice, J., & Balasubramanian, V. (2015). A principle of economy predicts the functional architecture of grid cells. *eLife*. doi:10.7554/eLife.08362

Wernle, T., Waaga, T., Mørreaunet, M., Treves, A., Moser, M.-B., & Moser, E. I. (2017). Integration of grid maps in merged environments. *Nature Neuroscience*. doi:10.1038/s41593-017-0036-6

Whittington, J., Muller, T., Mark, S., Barry, C., & Behrens, T. (2018). Generalisation of structural knowledge in the hippocampal-entorhinal system. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems 31* (pp. 8493–8504). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/8068-generalisation-of-structural-knowledge-in-the-hippocampal-entorhinal-system.pdf

Wills, T. J., Cacucci, F., Burgess, N., & O'Keefe, J. (2010). Development of the hippocampal cognitive map in preweanling rats. *Science*, *328*(5985), 1573–1576.

Witkin, A. P. (1983). Scale-space filtering. In *Proceedings of the eighth international joint conference on artificial intelligence - volume 2* (pp. 1019–1022). IJCAI'83. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Retrieved from http://dl.acm.org/citation.cfm?id=1623516.1623607

Young, R. A. (1987). The Gaussian derivative model for spatial vision: I. Retinal mechanisms. *Spat Vis*, *2*(4), 273–293.

Zilli, E. (2012). Models of grid cell spatial firing published 2005-2011. *Frontiers in Neural Circuits*, *6*, 16. doi:10.3389/fncir.2012.00016

A Algorithms

The following pseudo-code uses member-notation similar to common programming languages in the form σ.parents, where `parents` is a member of symbol σ. Also, multiple assignment instructions may be stated in a single line (e.g., Algorithm 4, line 2). A python implementation of the algorithms and additional code for visualization can be found at https://github.com/rochus/scalespacemts.

Algorithm 1 computes the expansion of a set of symbols. That is, given a set of transitions Γ and currently active symbols S_t at time t it computes all subsequent symbols of Σ that can be reached and have not been expanded previously. The algorithm also updates the parents of all such newly activated symbols. A transition γ has members `image` and `domain`, which are sets containing the symbols in the image and domain of the transition.

Algorithm 1 Expanding a graph to all subsequent symbols

```
procedure EXPAND(Γ, S_t, t)
    E ← ∅
    for all γ ∈ Γ do
        D_active ← S_t ∩ γ.domain
        if D_active ≠ ∅ then
            E ← E ∪ γ.image
            for all σ ∈ γ.image do
                if σ.parents = ∅ or σ.t_parents = t then
                    σ.parents ← σ.parents ∪ D_active
                    σ.t_parents ← t
                end if
            end for
        end if
    end for
    return E
end procedure
```

Given a set of transitions Γ, a set of symbols Σ, a set of starting symbols Σ_{start}, and a set of target symbols Σ_{target}, Algorithm 2 computes a sequence from any of the start symbols to any of the target symbols on one scale of transitions. The algorithm does so by calling Algorithm 1 repeatedly until the target symbol is in the set of currently active symbols.

Backtracking a solution sequence from target to start is computed as specified in Algorithm 3. That is, given a set of symbols Σ, a set of starting symbols Σ_{start} and a set of target symbols Σ_{target}, a path is computed by repeatedly randomly selecting a parent node until one starting symbol is found. Note that this requires that Algorithm 1 was applied to the symbols, whereby the parent field of each symbol is set accordingly.

Algorithm 4 takes a scale-space MTS $\mathcal{M}$, a set of symbols Σ, a set of startsymbols Σ_{start}, a set of target symbols Σ_{target}, and a number of maximal expansions per scale. It then computes at most i_{max} expansion according to Algorithm 1 per scale. If this maximum number is reached, the algorithm escalates to the next scale. $\mathcal{M}.\Gamma[s]$ denotes transition scale s of $\mathcal{M}$. Afterwards, Algorithm 3 can be used to compute a Monte Carlo solution sample.

Algorithm 2 Find a sequence on one scale of transitions with flood fill

```
procedure FindSequenceOnScale(Γ, Σ, Σ_start, Σ_target)
    t ← 0
    S_t, T ← Σ_start, Σ_target
    while S_t ∩ T = ∅ do
        for all σ ∈ S_active do
            σ.t_retrieval ← t
        end for
        N ← Expand(M.Γ[s], S_t, t)
        S_{t+1} ← {σ | σ ∈ N, σ ∉ S_t, σ.t_retrieval < 0}
        t ← t + 1
    end while
    return S_t ∩ T
end procedure
```

Algorithm 3 Backtrack from target to start

```
procedure Backtrack(Σ, Σ_start, Σ_target)
    S ← Σ_target
    P ← {σ_p | σ_p ∈ σ_s.parents, σ_s ∈ S}
    while Σ_start ∩ P = ∅ do
        S, P ← P, {σ_p | σ_p ∈ σ_s.parents, σ_s ∈ S}
    end while
    return S, P
end procedure
```

Algorithm 4 Find a sequence of symbols by ascending the scale-space

```
procedure FindSequenceAscending(M, Σ, Σ_start, Σ_target, i_max)
    i, t, s ← 0, 0, 0
    S_t, T ← σ_start, σ_target
    while S_t ∩ T = ∅ do
        for all σ ∈ S_active do
            σ.t_retrieval ← t
        end for
        N ← Expand(M.Γ[s], S_t, t)
        S_{t+1} ← {σ | σ ∈ N, σ ∉ S_t, σ.t_retrieval < 0}
        if i ≥ i_max then
            s, i ← min{s + 1, N_scales}, 0
        end if
        t, i ← t + 1, i + 1
    end while
end procedure
```

Algorithm 5 takes the same arguments as Algorithm 4 except the maximal number of scales. The algorithm starts on the highest available transition scale and finds a sequence with it. After finding a solution, it backtracks to the starting symbols and generates subgoals. Subsequently, it drops down one scale to refine the trajectory to the subgoals. Note that in its current form this requires to reset the parent information in each symbol (line 6).

Algorithm 5 Find a sequence of symbols by descending the scale-space

```
1:  procedure FindSequenceDescending(M, Σ, Σ_start, Σ_target)
2:      s ← N_scales − 1
3:      T ← Σ_target
4:      while s ≥ 0 do
5:          for all σ ∈ Σ do
6:              σ.t_retrieval, σ.t_parents, σ.parents ← −1, −1, ∅
7:          end for
8:          H ← FindSequenceOnScale(M.Γ[s], Σ, Σ_start, T)
9:          T, P ← Backtrack(Σ, Σ_start, H)
10:         s ← s − 1
11:     end while
12: end procedure
```