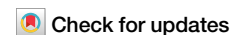


<https://doi.org/10.1038/s44335-026-00052-w>

Brain Inspired Probabilistic Occupancy Grid Mapping with Vector Symbolic Architectures

Shay Snyder¹ ✉, Andrew Capodieci², David Gorsich³ & Maryam Parsa¹ ✉

Real-time robotic systems require advanced perception and action capability. However, the main bottleneck in current autonomous systems is the trade-off between computational capability, energy efficiency, and model determinism. World modeling, a key objective of many robotic systems, commonly uses occupancy grid mapping (OGM) as the first step towards building an end-to-end robotic system. OGM discretizes the environment into cells and assigns probability values to attributes such as occupancy. Existing methods fall into two categories: traditional methods and neural methods. Traditional methods leverage dense statistical calculations, while neural methods employ deep learning for probabilistic information processing. We propose a vector symbolic architecture-based OGM system (VSA-OGM) that retains the interpretability and stability of traditional methods with the improved computational efficiency of neural methods. VSA-OGM, validated across multiple datasets, achieves similar accuracy to covariant traditional methods while reducing latency and memory by 45× and 400×, respectively. Compared to invariant traditional methods, VSA-OGM maintains similar accuracy values while reducing latency by 5.5×. Moreover, VSA-OGM achieves 6× latency reductions compared to neural methods while eliminating the need for domain-specific training. This work demonstrates the potential of VSA-OGM as a foundation for efficient OGM in autonomous systems operating under strict computational and latency constraints.

An essential capability for real-time robotic systems is the accurate and computationally efficient probabilistic mapping of environments. Efficient probabilistic mapping is the central idea within OGM where environments are discretized into individual cells, also known as pixels and voxels in 2D and 3D environments, respectively¹. Each cell is then assigned labels or properties such as occupancy, traversability, or other relevant attributes based on application-specific requirements². OGM methods are broadly categorized into two main areas: traditional and neural methods. Traditional methods^{3–11} adopt a dense approach where each cell is assigned a fully defined probability distribution or feature representation.

In contrast, neural methods, such as^{12–19} utilize deep learning architectures, such as convolutional neural networks^{12,17}, neural radiance fields¹⁸, and large language models¹⁹, to create a black-box solution. This approach mitigates many of the computational challenges associated with traditional methods but introduces fragility to domain adaptation and model stochasticity through domain-specific pretraining with backpropagation²⁰. Although neural methods present computational advantages over traditional approaches, their intrinsic stochasticity and

lack of interpretability pose significant validation challenges, particularly in safety-critical applications²¹. Moreover, recent neural methods leverage foundation models to perform probabilistic inference, making them unsuitable for real-time deployment on edge devices with computational and power constraints¹⁹.

Recent advances in cognitive science^{22,23} and computational neuroscience^{24,25} highlight the unique computational properties of hyperdimensional spaces. Also known as vector symbolic architectures (VSA), these methods are actively being explored as mathematical analogs to the higher-order computational capabilities of spiking neurons^{25,26}. Furthermore, VSA approaches possess unique computational properties allowing them to statistically, and deterministically, encode semantic and spatial information^{25,27,28}.

In this work, we introduce a VSA framework, VSA-OGM, leveraging the unique probabilistic properties of bio-inspired hyperdimensional vectors to efficiently and accurately perform OGM. Our approach draws inspiration from both traditional³ and neural¹² methods, aiming to strike a balance between the accuracy, interpretability, stability, and domain

¹George Mason University, Fairfax, VA, USA. ²Neya Systems, LLC, Warrendale, PA, USA. ³U.S. Army Transformation and Training Command, Warren, MI, USA.

✉ e-mail: ssnyder9@gmu.edu; mparsa@gmu.edu

Table 1 | A summary of all single-agent datasets used in this paper: Ablation (Dense), Ablation (Sparse), ToySim⁶, Intel Map²⁹, and EviLOG¹²

Dataset	Type	Map Size	Points/Scan	# Scans	% Occupied	% Empty	Density
Ablation (Dense)	Simulated	36 m ²	100	100	50%	50%	277.8
Ablation (Sparse)	Simulated	36 m ²	100	10	50%	50%	27.8
ToySim ⁶	Simulated	10,000 m ²	50	90	60%	40%	0.45
Intel Map ²⁹	Real	1440 m ²	421	910	58%	42%	266
EviLOG ¹²	Simulated	4614 m ²	48000	11100	–	–	–

Dataset density is defined as the total number of labeled points divided by the environment area (points/m²). (–) Points are provided without labels.

Table 2 | A summary of all multi-agent datasets used in this paper: ToySim⁶ and Intel Map²⁹

Dataset	Agent Id	Map size	Points/Scan	# Scans	% Occupied	% Empty
ToySim ⁶	1	10,000 m ²	50	367	48.9%	51.1%
	2				53.1%	46.9%
Intel Map ²⁹	1	1440 m ²	421	222	38.8%	61.2%
	2			161	43.4%	56.6%
	3			283	42.3%	57.7%
	4			244	42.1%	57.9%

Table 3 | A summary of the VSA-OGM parameters used in all experiments

Experiment	d	l	δ	$r(\psi = 0)$	$r(\psi = 1)$	Multi-Agent?
ToySim ⁶	32,000	1.0	2	3	3	$\times$
						$\checkmark$
Intel Map ²⁹	32,000	0.2	8	1	1	$\times$
						$\checkmark$
EviLOG ¹²	32,000	0.2	1	3	3	$\times$

flexibility of traditional methods with the computational efficiency of neural methods. We refer to this intersection of traditional and neural methods as neuro-symbolic methods.

To the best of our knowledge, VSA-OGM is the first large-scale application and development of a VSA framework for real-time OGM. Throughout the remainder of this work, we discuss the specifics of VSA-OGM and highlight the performance of our framework on five synthetic datasets^{6,12} and one real dataset²⁹ compared to multiple traditional methods^{4,6,8–10}, and one neural method¹². Additional information on OGMs and VSAs can be found in “Methods”. We also conduct an in-depth ablation study providing further insight into the capabilities and unique aspects of VSA-OGM. We conclude with a detailed discussion of our results and potential avenues for future research. In summary, the major contributions of our paper are:

- We introduce a neuro-symbolic VSA framework for OGM, VSA-OGM, that maintains the accuracy of traditional methods while leveraging the improved computational characteristics of neural methods.
- Compared to traditional covariant methods that explicitly maintain covariance between cells^{6,9}, across two simulated⁶ and one real-world²⁹ dataset, VSA-OGM achieves approximately 400x and 45x reductions in memory usage and latency, respectively, while maintaining comparable accuracy.
- We demonstrate the single- and multi-agent mapping capabilities of VSA-OGM compared to two invariant traditional methods^{4,8}, which do not maintain covariance between cells, achieving approximately 5.5x lower latency while preserving comparable accuracy.

- We highlight the capabilities of VSA-OGM, which achieves approximately 6x lower latency than neural methods¹², without requiring model pretraining and maintaining comparable mapping performance.
- We conduct a comprehensive ablation study to evaluate the performance impact of various hyperparameter and framework configurations in VSA-OGM. Our results clarify the role of each component and reveal how their integration forms a neurosymbolic framework for OGM. This enables VSA-OGM to retain the accuracy and determinism of traditional methods while surpassing the computational efficiency of neural methods, without requiring extensive pretraining or introducing domain fragility.

Results

To evaluate the performance of VSA-OGM, we use five datasets and nine baseline methods. The first dataset was synthetically created with the simulator from Bayesian Hilbert Maps⁶ (*ToySim*). The second is the Intel Campus dataset that was captured with a physical robot at Intel’s Seattle campus²⁹ (*Intel Map*). The third is the simulated autonomous driving dataset from Evidential LiDAR Occupancy Grid Mapping (*EviLOG*)¹² (*EviLOG*). The fourth and fifth are simplified datasets we created for our ablation study with dense (*AblationDense*) and sparse (*AblationSparse*) variants. A summary of these datasets is shown in Table 1 with all multi-agent variants shown in Table 2.

Mapping accuracy is measured as the area under the receiver operating characteristic curve (AUC)³⁰ and the negative log-likelihood (NLL). All experiments are conducted using a desktop workstation with an Intel(R) Xeon(R) W-2295 CPU, 128GB RAM, and an RTX A5000 GPU.

We compare VSA-OGM with 8 traditional methods and one neural method. Since most existing neural OGM methods are designed for 3D environments^{2,31,32}, we restrict our comparison to a 2D neural baseline to ensure a fair and direct evaluation against our 2D approach. A summary of all parameters used in each experiment is shown in Table 3. An ablation study of all hyperparameters and factors influencing VSA-OGM’s performance is provided in Supplementary Note: Ablation Study. Below, we summarize the primary design choices for each dataset that highlight VSA-OGM’s stability over a range of hyperparameters. We split our experiments into Single-Agent and Multi-Agent tasks, where the former uses a single agent to map the entire

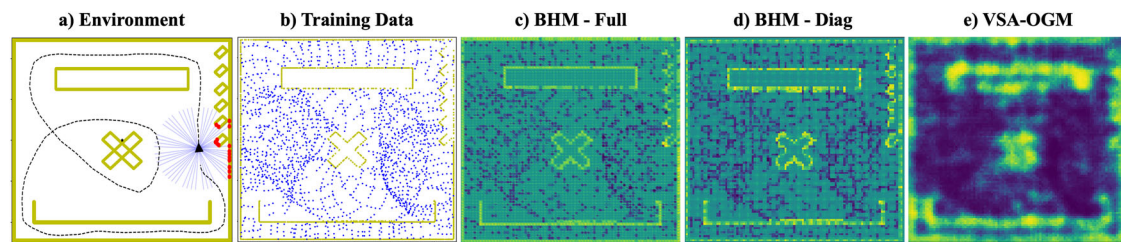


Fig. 1 | Single-Agent Mapping with ToySim⁶. **a** The simulated testing environment with the agent's trajectory. **b** The sparse point cloud data with ground truth labels for training. **c** Trained OGM from BHM with a fully-defined covariance matrix.

d Trained OGM from BHM with a diagonal covariance matrix. **e** Trained OGM from VSA-OGM.

Table 4 | A comparison between VSA-OGM and Bayesian Hilbert Maps (BHM)⁷, with and without a fully defined covariance matrix, on the simulated dataset created from our reimplementation of the simulator introduced in ref. 6

Method	AUC	NLL	CPU latency	GPU latency	Model size
BHM – Full ⁶	0.96	0.44	6.5 s	–	6400 MB
BHM – Diag ⁶	0.97	0.38	0.45 s	0.024 s	0.04 MB
VSA-OGM (ours)	0.93	0.33	0.04 s	0.005 s	1.02 MB

(–) Open-source implementation is unavailable.

environment and the latter uses multiple agents whose individual perspectives are combined to create a single “fused” OGM.

ToySim—Single-Agent

We created a synthetic LiDAR dataset with a re-implementation of the simulator introduced in BHM⁷. This dataset uses a single agent equipped with a single LiDAR sensor having a 360° field of view, a view distance of 20 m, and 50 individual beams. The total area of the environment is 10,000 m². The dataset contains 4500 individual points with a label distribution of 60% occupied and 40% empty. The dataset statistics are also summarized in Table 1.

We train VSA-OGM and BHM⁶ on 80% of the dataset with 20% reserved for validation where the final AUC scores and negative log likelihood values are determined. This dataset is particularly difficult due to the sparse features. Many traditional applications of probabilistic OGM avoid this issue by augmenting the training data. This is accomplished by interpolating a linear line with random points between the agent's location and the LiDAR points⁶. We chose to avoid this preprocessing step to highlight how traditional methods and VSA-OGM respond to data sparsity.

With a vector dimensionality of 32,000, a length scale of 1.0, occupied and empty disk filter radii of 3 cells, and 2 tiles per dimension, VSA-OGM achieves an AUC score of 0.93 versus BHM with scores of 0.96 and 0.97 for the versions with a full and diagonal covariance matrix, respectively. BHM – Full maintains covariance between all environmental cells, whereas BHM – Diag discards covariance (retaining only per-cell variances). Comparing the resulting NLL scores, VSA-OGM has the lowest value of 0.33 compared to BHM – Diag with a value of 0.38 and BHM – Full with a value of 0.44. Although the simultaneous improvement in NLL and reduction in AUC may appear counterintuitive, this indicates that VSA-OGM is less effective at ranking data points but produces probability estimates that are better calibrated to the true values. This characteristic of VSA-OGM is a result of the utilization of a softmax function to convert the quasi-probability values into true values. Refer to Eq. (13) and Fig. 8 for more information.

Despite adjusting the kernel-width parameter for either BHM implementation, we were unable to build an OGM that successfully interpolates between sparse points to broadly classify empty regions with no training or testing data. Therefore, all results presented in this section

use the same hyperparameters as specified in Table 3. This is shown in Fig. 1 where the majority of the map is labeled as unknown, highlighted by the teal color. On the other hand, VSA-OGM was able to successfully interpolate between the empty points and label the majority of the area as empty. Our increased interpolation performance also comes with significantly reduced inference latency at approximately 162× reductions versus BHM – Full (CPU) while maintaining comparable accuracy on the training and validation sets. VSA-OGM reduces the total memory size of the learned representation by 6400×. The GPU latency numbers for BHM – Full are omitted because we were unable to find a working GPU-compatible implementation. Compared to BHM – Diag, VSA-OGM provides latency reductions of approximately 10× (CPU) and 5× (GPU) at the expense of increased memory complexity. However, we still see the same issue where neither version of BHM is able to interpolate between the sparsely observed regions of the environment. This is shown visually in Fig. 1c, d, where both BHM variants produces sharp, high-frequency peaks that closely follow the training data but provide little interpolation in sparse regions. In contrast, VSA-OGM generates coarser, lower-frequency features that, while less detailed, interpolate smoothly across unlabeled regions with reduced latency. These results are summarized in Table 4. The labeled points and trained OGMs are visualized in Fig. 1.

ToySim—Multi-Agent

We created a second dataset from our re-implemented simulation environment from ref. 7. Rather than being focused on a single agent, this experiment simulates the accumulation of data from two independent agents operating within the same environment. As such, each will have a unique perspective of the environment that should help improve the collective world model. The same configuration parameters are used from Table 3. Therefore, both agents are equipped with identically configured hyperdimensional mapping systems. Both agent's are trained simultaneously on 80% of their training data along with fusion occurring at every time step. The AUC score is calculated across both agent's individually with Agent 1 achieving an AUC score of 0.93 and a NLL of 0.32. Agent 2 achieves an AUC score of 0.96 and a NLL of 0.46. Furthermore, the final fused map achieves an AUC score of 0.95 and a NLL of 0.32, suggesting that our fusion methodology results in negligible information losses and produces a global representation with high AUC values and low NLL scores. The individual training results amongst both agents are shown in Fig. 2. The fusion operation is approximately equivalent to processing one point cloud and generating the map at 5 ms. These performance and latency numbers for each agent and the fused representation are shown in Table 5.

Intel Map—Single-Agent

The Intel Map dataset²⁹ is a commonly used, real-world dataset for benchmarking OGM algorithms. Using a vector dimensionality of 32,000, a length scale of 0.2 m, disk filter radii of 1 cell, and 8 quadrants per dimension, VSA-OGM (CPU) achieves an AUC score of 0.95 with an average per point cloud

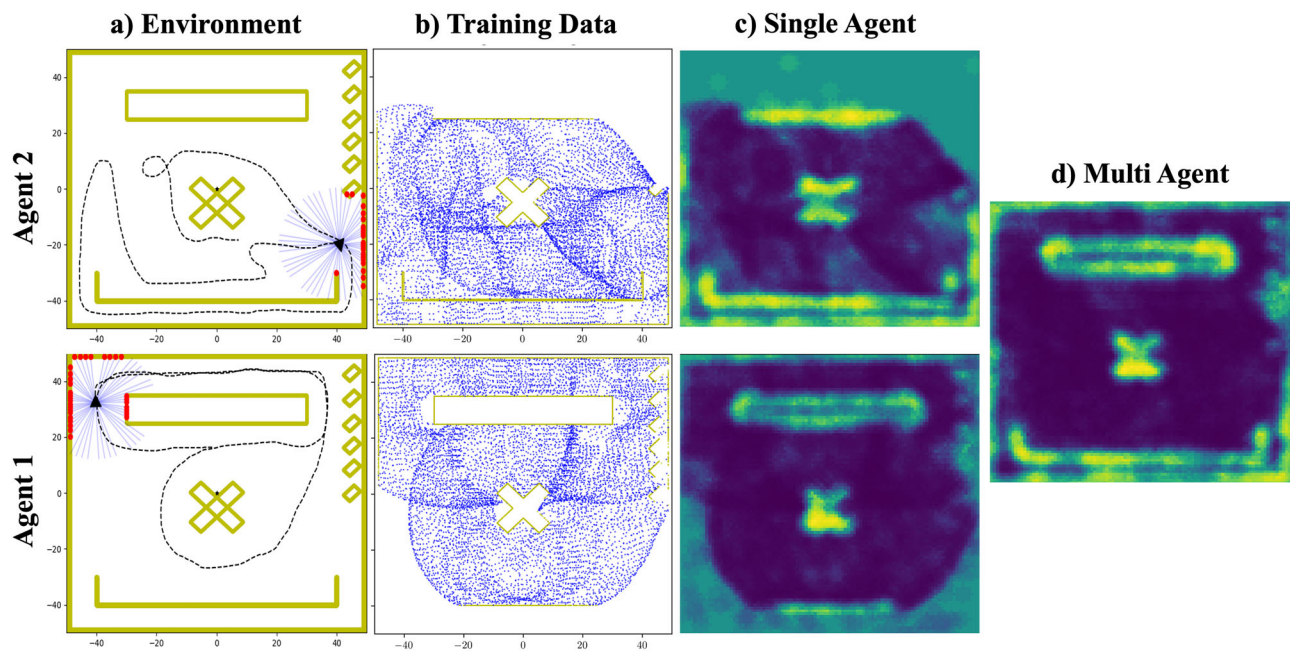


Fig. 2 | Multi-Agent Mapping with ToySim⁶. **a** The simulated testing environments with both agent's unique trajectories. **b** The sparse point cloud data with ground truth labels for training. **c** The global OGM from VSA-OGM across each individual

agent. **d** The combined OGM from VSA-OGM with the combined knowledge from Agent 1 and Agent 2.

Table 5 | Multi-agent mapping with the ToySim dataset and VSA-OGM

Agent	AUC	NLL	Latency (s)
Agent 1	0.93	0.32	0.005
Agent 2	0.96	0.46	0.005
Map fusion	0.95	0.32	0.006

The AUC score³⁰, Negative Log-Likelihood (NLL), and the per point cloud processing latency of agents 1 and 2 on our simulated dataset before being merged into a single, fused representation.

processing time of 510 ms versus BHM, with a full covariance matrix (BHM – Full)⁶, with a score of 0.96 and a processing time of 22.35 s. Furthermore, BHM with a diagonal covariance matrix (BHM – Diag) achieves a final AUC of 0.96 with a processing time of 8.11 s. These results are summarized in Table 6 with the ground truth representations and trained OGMs visualized in Fig. 3.

We were unable to compare against BHM – Full on the GPU because the open-source implementation does not support GPU execution. BHM – Diag does support GPU execution, and VSA-OGM achieves an order-of-magnitude latency reduction at a spatial resolution of 0.2 m. However, we could not scale BHM – Diag to a resolution of 0.1 m because its dynamic memory requirements exceeded the available 24GB of GPU memory. In terms of memory, VSA-OGM provides an approximate 392× reduction compared to the BHM – Full. No memory reduction is achieved with respect to BHM – Diag because this method assumes independence between cells, which drastically lowers the memory footprint.

Relative to Fast-BHM⁴, VSA-OGM reduces per point cloud latency by an order of magnitude while maintaining covariance between cells. Unfortunately, a GPU-compatible implementation of Fast-BHM was unavailable, and the runtimes reported in the original paper were unclear as to whether they included both map generation and training. If map generation was included (as in our BHM experiments) its latency would likely be substantially higher than the reported numbers. Since Fast-BHM assumes independence between voxels, no memory reduction is seen

compared to VSA-OGM. As shown in Table 6, VSA-OGM offers one to two orders of magnitude latency reductions compared to earlier traditional methods^{5,9}. Compared to a more recent traditional approach⁸, VSA-OGM reduces CPU latency by approximately 2×.

VSA-OGM establishes a new paradigm in occupancy grid mapping (OGM): a neurosymbolic approach that is faster, more memory-efficient, and extensible to future applications. However, analyzing both the resulting maps and runtime configurations, two fundamental limitations of VSA-OGM emerge:

1. Although the ToySim experiments suggest interpolation across sparse regions, performance degrades when data is highly imbalanced (e.g., dense hallways versus sparse rooms). This highlights sensitivity to non-uniform spatial sampling.
2. As point counts increase by orders of magnitude, memory vectors become saturated, forcing us to increase the number of tiles per dimension (δ) to maintain accuracy. Thus, vector dimensionality must scale with dataset size, or the number of tiles per dimension must be increased accordingly. This is a limitation we would like to address in future works by developing an information theoretic method to maintain consistent mapping performance while avoiding memory saturation.

These limitations highlight open challenges within VSA-OGM, such as balancing dimensionality with information density. We will develop information-theoretic strategies for adaptive dimensionality allocation, ensuring that VSA-OGM preserves accuracy without saturating memory vectors.

Intel Map – Multi-Agent

Based on the multi-map fusion technique described in ref. 4, we split the Intel Map dataset²⁹ into 4 quadrants across the x – y plane. The splits of the global map are highlighted in Fig. 4. A quantitative summary of the data is shown in Table 2. As shown in Table 3, all agents are equipped with identically configured hyperdimensional mapping systems. Therefore, each system is initialized with a vector dimensionality of 32,000, a length scale of 0.2 m, disk radii of 1, and 8 quadrants per dimension. As shown in Table 7,

Table 6 | Comparison between VSA-OGM, OGM⁵, I-GPOM⁹, I-GPOM2⁹, SBHM⁶, Fast-BHM⁴, and SBKM⁸ on the Intel Map dataset²⁹ (space delta) the resolution of the decoded OGM in meters per cell

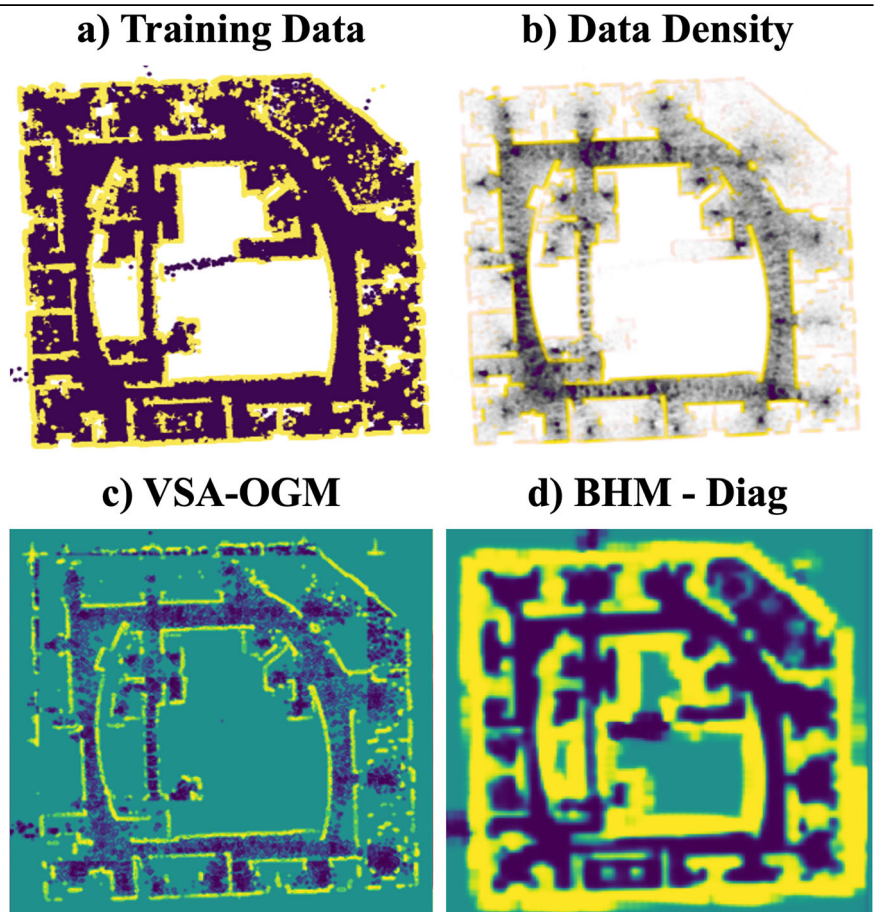
Parameters	γ	Σ	# features	space delta	AUC	NLL	Model Size	CPU Time	GPU Time
OGM ⁵	–	–	–	◦	0.93	◦	◦	1.12 s	–
I-GPOM ⁹	3/2	full	–	◦	0.94	◦	◦	15.35 s	–
I-GPOM2 ⁹	3/2	full	–	◦	0.97	◦	◦	17.17 s	–
BHM ⁶	6.71	full	5600	0.1	0.96	◦	6400MB	22.35 s	–
	6.71	diag		0.1	0.96 ^a	0.25 ^a	0.04 MB	8.11 s ^a	X ^a
	6.71	full		0.2	0.98	0.24	6400 MB	11.8 s ^a	–
	6.71	diag		0.2	0.98	0.24	0.04 MB	2.11 s ^a	0.11 s ^a
Fast-BHM ⁴	◦	diag	5600	0.1	0.95	◦	0.04 MB	0.46 s ^b	–
SBKM ⁸	6.71	full	3492	0.2	0.96	0.36	◦	0.76 s ^b	–
	6.71	$\lambda_{\max}$		0.2	0.95	0.52	◦	0.43 s ^b	–
VSA-OGM	–	–	32000	0.1	0.95	0.25	16.3MB	0.51 s	0.04 s
	–	–		0.2	0.95	0.25		0.26 s	0.02 s

^aAdditional experiments we performed with the baseline methods. (X) unable to evaluate due to excessive GPU memory usage. (◦) values are not listed in original manuscript.

^bOriginal manuscript did not specify if latency values include point cloud processing and map generation.

Fig. 3 | Single-Agent Mapping with Intel Map²⁹.

a The ground-truth labeled training data. **b** The density of the training data. **c** The global OGM from VSA-OGM. **d** The final OGM from BHM with a diagonal covariance matrix.



the final AUC values on the 20% validation set were 0.95, 0.96, 0.94, and 0.93. Furthermore, the global fused AUC was 0.95 compared to Fast-BHM⁴ with a final AUC of 0.94. Our results also include an experiment performed in Fast-BHM⁴, where they compare against another method known as Overlapping Hilbert Maps (OHM)¹⁰. Compared to OHM, VSA-OGM improves the fused AUC by 0.03. Therefore VSA-OGM achieves improved performance to Fast-BHM and OHM in multi-agent fusion. These values, as

well as the per quadrant AUC and NLL values, are summarized in Table 8 with the training data and OGMs visualized in Fig. 4.

EviLOG Dataset

Introduced in ref. 12, Evidential LiDAR Occupancy Grid Mapping (EviLOG) uses a multi-layer convolutional neural network with 4.7 million

Fig. 4 | Multi-Agent Mapping with Intel Map²⁹.
a The four quadrant occupancy grid maps with VSA-OGM on the Intel Map dataset²⁹. **b** The density of the training data. **c** The global combined OGM from VSA-OGM. **d** The final OGM from Fast-BHM⁴.

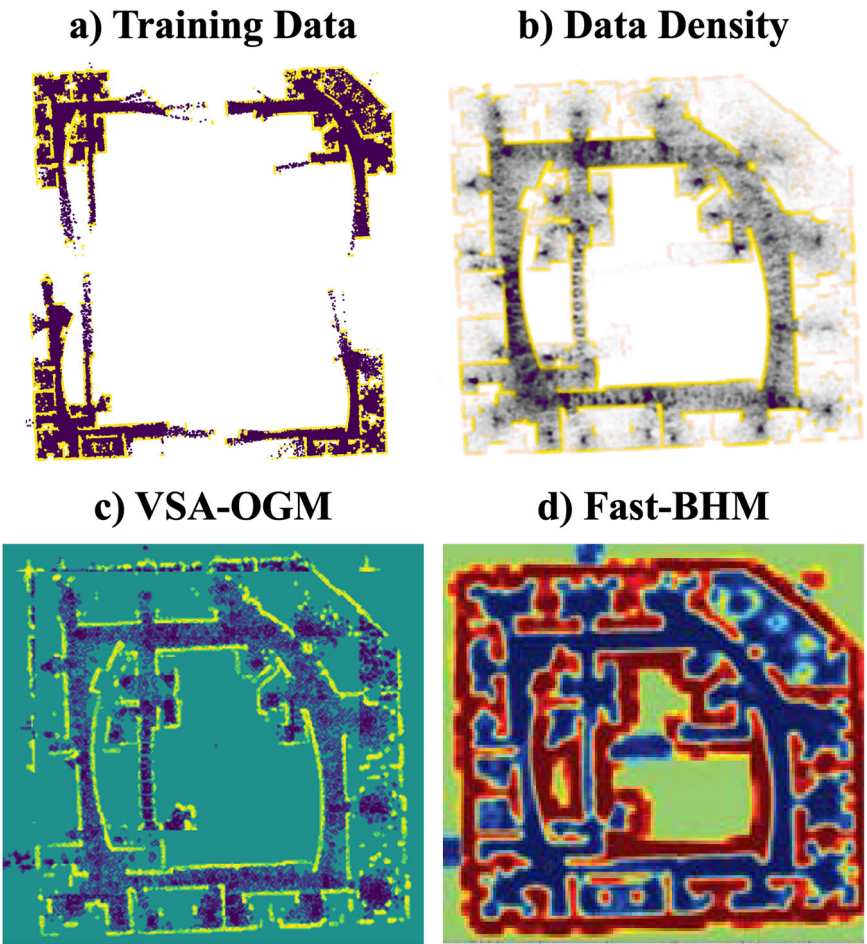


Table 7 | The performance of the individual VSA-OGM agents on the multi-agent Intel Map²⁹ dataset

Quad.	AUC	NLL	Latency (s)
1	0.95	0.45	0.04
2	0.96	0.39	0.04
3	0.94	0.42	0.04
4	0.93	0.37	0.04
Fusion	0.95	0.37	0.03

trainable parameters. Since they introduce a dataset and a network architecture, we refer to their dataset as EviLOG dataset and their baseline network as EviLOG-BaseNet. They introduce a fully-labeled synthetic OGM dataset using a VLP32C LiDAR sensor mounted on the roof of a moving vehicle. The EviLOG dataset is separated into three sets for training, testing, and validation with each containing 10,000, 100, and 1000 point clouds, respectively. Each point cloud is bounded within a 56.32 by 81.92 m rectangular region. Furthermore, each point cloud has a corresponding ground truth OGM with a cell resolution of 0.16 m.

Training EviLOG-BaseNet for 100 epochs, with a batch size of 5, took approximately 30 h. This comes in contrast to VSA-OGM where no pre-training is required and represents a limitation of neural methods. We created a small variant of EviLOG-BaseNet known as EviLOG-SmallNet. By reducing the number of features in the convolution layers by 8x, EviLOG-SmallNet has 76K model compared to VSA-OGM with 64K parameters. We performed the same training process with EviLOG-SmallNet and this process took approximately 13 h.

Table 8 | The multi-map fusion capabilities of VSA-OGM versus Fast Bayesian Hilbert Maps⁴ and Overlapping Hilbert Maps¹⁰ on the Intel Map dataset²⁹

Metric	VSA-OGM	Fast-BHM ⁴	OHM ¹⁰
AUC	0.95	0.936	0.92
Model size	16.4 MB	0.04 MB	–
Covariance	True	False	False

The reduction of model size within⁴ is due to their approach assuming independence between cells. (–) Values are not provided in the original manuscript.

VSA-OGM has a per frame inference time of 55 ms versus the EviLOG architectures at 165 ms and 290 ms for the small and original variants respectively. Therefore without pretraining, VSA-OGM is approximately 3× and 6× faster with a reduction in model parameters by 16% and 99%, respectively. The configuration of VSA-OGM utilized in this experiment is shown in Table 3. A summary of each model configuration, training latency, and inference latency is shown in Table 9.

EviLOG-BaseNet calculates performance based on the KL-divergence across the entire domain, including areas with no labeled points, so we are unable to compare via quantitative analysis with AUC and NLL. Therefore, we leverage qualitative analysis between the resulting OGMs.

The EviLOG architectures begin with a PointPillars backbone³³, before passing the latent representation through a series of convolution layers. This allows both EviLOG architectures to infer structure beyond directly observed points, leveraging priors learned during training to extrapolate into completely unobserved regions. This comes in contrast to VSA-OGM which makes no prior assumptions about the observable space but uses the

point labeling heuristic defined in the EviLOG paper¹². As shown in Fig. 5, VSA-OGM produces accurate occupancy estimates in regions where observations exist but remain spatially sparse, however it struggles in areas that are completely unobserved. This distinction suggests that VSA-OGM is well suited for partially observed settings, while the EviLOG architectures are more effective when strong priors are needed to extrapolate structure beyond the observed domain.

Discussion

We present an approach to OGM drawing inspiration from recent advances in cognitive science^{22,25} and computational neuroscience²⁴. Our methodology is grounded in the evolving field of vector symbolic architectures (VSA)³⁴. VSAs have been conceptualized as a computationally viable abstraction for modeling biological systems, including memory, cognition, and motor control²⁴. To the best of our knowledge, this work is the first to apply VSAs to the construction of incremental, probabilistic OGMs suitable for real-world environments.

Table 9 | Comparing the training time and inference latency of VSA-OGM against EviLOG-BaseNet¹² and EviLOG-SmallNet

Model	Model size	Training time	Inference latency
EviLOG-BaseNet	4.7M	30 h	290 ms
EviLOG-SmallNet	76K	13 h	165 ms
VSA-OGM	64K	–	55 ms

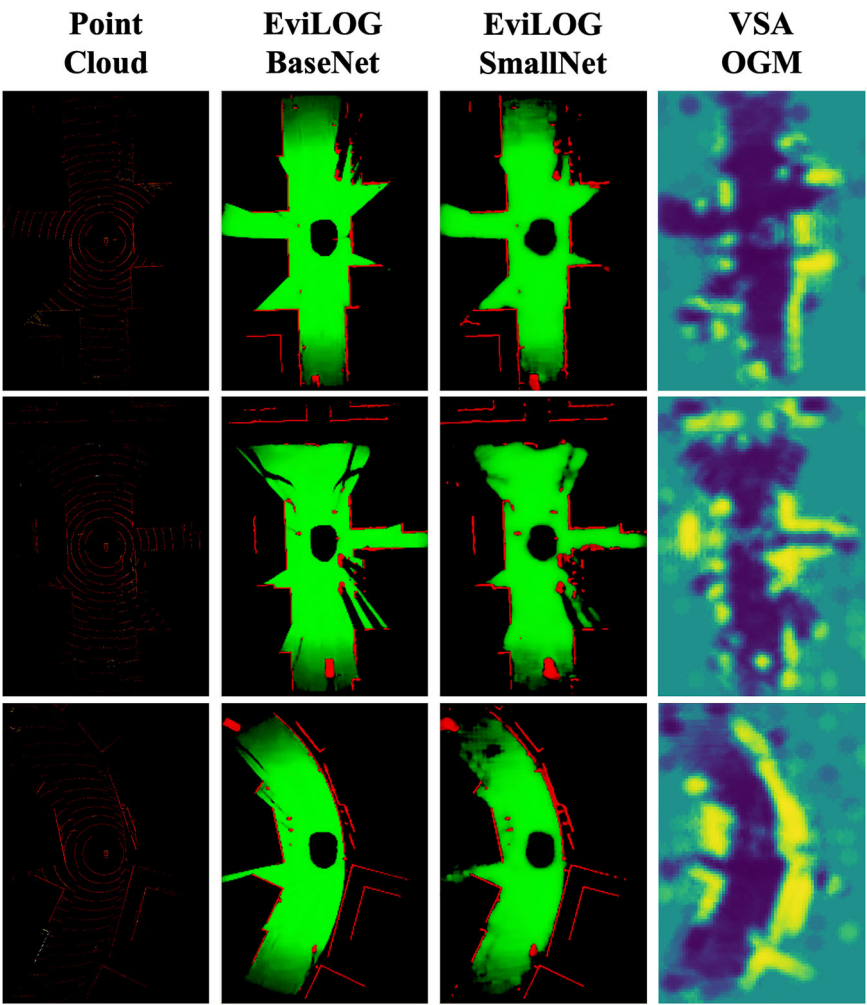
This research builds upon the work of²⁶ which introduced Spatial Semantic Pointers (SSP), an algebraic framework founded on real-valued hyperdimensional vectors and circular convolution. Previous studies demonstrate the efficacy of SSP-based VSA systems serving as kernel density estimators in hyperdimensional space³⁵. Expanding upon these prior works, we describe an end-to-end probabilistic OGM framework founded on SSPs, known as VSA-OGM.

Existing OGM approaches encounter two prominent challenges. Firstly, traditional methods offer high accuracy but struggle with significant time complexity due to the learning and parameterization of fully specified statistical distributions. Secondly, neural methods demonstrate high accuracy in extrapolating beyond observed domains with reduced latency but have limited application in mission and safety critical applications due to operating stochasticity in unseen domains. Moreover, neural methods are constrained by domain-specific pretraining which limits their application in novel domains without repeating the pre-training process.

In response to these challenges, VSA-OGM pioneers a novel direction in OGM systems. We refer to this new direction as neuro-symbolic methods since they draw inspiration from traditional and neural methods. Neuro-symbolic methods seek to leverage the determinism and high accuracy of traditional methods while harnessing the enhanced computational efficiency of neural methods. By amalgamating the strengths of both domains, neuro-symbolic methods offer a promising solution to address the computational complexity and stochasticity issues prevalent in current mapping techniques.

We showcase the efficacy of VSA-OGM as a neuro-symbolic method through comprehensive evaluation, including an ablation study, quantitative, and qualitative comparisons with state-of-the-art traditional^{4–6,8–10} and

Fig. 5 | A comparison of VSA-OGM with EviLOG-BaseNet¹² and EviLOG-SmallNet. The input point clouds and decoded OGMs from each approach are highlighted.



neural methods¹². In our ablation study, we assess the influence of our entropy-based decoding mechanism, its associated parameters, the environment discretization mechanism, and the vector length scale. Our findings illuminate the impact of these parameters on the accuracy and stability of system performance. Our results and analysis underscore the robustness and adaptability of VSA-OGM, positioning it as a promising alternative that bridges the strengths of both traditional and neural methods in addressing complex spatial mapping tasks.

Compared to the traditional methods Bayesian Hilbert Maps (BHM)⁶ and Fast Bayesian Hilbert Maps (Fast-BHM)⁴, VSA-OGM achieves similar levels of mapping accuracy while reducing inference latency by approximately 45× and 5.5×, respectively. Furthermore, VSA-OGM facilitates multi-map fusion among multiple agents. Our results demonstrate minimal information loss during the fusion process and performance comparable to that of Fast-BHM. This capability highlights the versatility and scalability of VSA-OGM in collaborative, multi-agent mapping scenarios.

In comparison to the neural method presented in EviLOG¹², VSA-OGM demonstrates the ability to accurately model and interpolate LiDAR points sampled along major road sections, and around moving vehicles without the need for pretraining sessions. This highlights the efficiency and effectiveness of VSA-OGM in capturing complex environmental dynamics. However, it's important to note a key distinction between VSA-OGM and EviLOG: while VSA-OGM excels in accurately representing data within the observed domain, it encounters challenges in extrapolating outside this domain, particularly in areas with extremely limited information. This limitation arises from VSA-OGM's fundamental principle of not assuming any specific properties about the operating environment. It is worth noting that EviLOG may be more susceptible to domain adaptation issues, as it is finely tuned to extrapolate domain-specific information.

Moving forward, our aim is to expand VSA-OGM to accommodate semantic 3D applications³⁶. Furthermore, we intend to explore strategies for mitigating the performance limitations associated with VSA-OGM's lack of assumptions about the environment and efficiently managing the limited memory vector capacity as data density increases.

Methods

We begin this section an overview of existing OGM approaches before discussing the mathematics for encoding statistical and spatial relationships within hyperdimensional space. Next, we define our hyperdimensional OGM framework, Vector Symbolic Architectures for Occupancy Grid Mapping (VSA-OGM), and discuss how point cloud data is encoded, processed, and subsequently decoded to create probabilistic OGMs.

Occupancy Grid Mapping (OGM)

OGM discretizes continuous, n -dimensional environments into discrete cells, commonly known as pixels or voxels in 2D and 3D environments, respectively². Cells are assigned arrays of attributes, ranging from binary indicators of occupancy to more complex attributes, such as cost values representing the likelihood of traversal through a region. OGM has diversified into several distinct approaches, with each having its own operational characteristics. *Traditional methods*, such as those described in refs. 3,5, represent the environment using dense grids of statistical distributions, where each cell explicitly models occupancy probabilities. More recent traditional methods, such as^{4,6,7}, project probabilistic information into a continuous Hilbert space to improve computational and space complexity. In contrast, *neural methods*^{12,17,31,37} leverage deep learning to encode environmental information into network weights, where probabilistic relationships are captured implicitly through hidden representations.

Traditional OGM methods

Traditional methods, introduced by Elfes⁵, rely on a dense, grid-based discretizations of the environment, where cells store probabilistic estimates of occupancy. These methods use geometric sensor models with Bayesian updates to recursively estimate occupancy from sensor data, providing a deterministic and interpretable map representation. However, the grid's

fixed resolution can limit accuracy and scalability, especially in large or complex environments, since computational and memory complexity scales linearly with the number of cells.

To overcome the resolution limitations of discrete grids, continuous OGM methods have been developed, which approximate occupancy as a spatially continuous function rather than discrete cells. Among these, Gaussian Process (GP)-based OGM is a well-established approach that models spatial correlations and uncertainty directly from sensor data⁹. While GPs provide smooth and high-fidelity occupancy estimates, their computational complexity grows cubically with the number of observations, versus the size of the environment, making them challenging to apply directly in real-time settings with high-fidelity sensors (i.e., high density point clouds)⁹.

Bayesian Hilbert Maps (BHM)s⁶ were introduced as an alternative to GP-based OGM approaches that represent occupancy as projections onto a sparse set of basis functions in Hilbert space. BHM's avoid storing explicit observations and reduce runtime and memory costs significantly while maintaining probabilistic interpretability. Fast-BHM⁴ further improves computational performance, to sub-quadratic levels, by assuming independence between basis functions to further approximate the Bayesian update.

More recent works such as, Sparse Bayesian Kernel-based OGM (SBKM)⁸ extend this line of work by focusing on sparse structural features within the environment, such as lines and curves, rather than dense occupancy estimates. This makes SBKM especially useful for efficiently capturing high-level geometry relevant to tasks like path planning and map simplification, though it is not intended as a direct replacement for dense occupancy grids⁸.

Neural OGM methods

Neural methods, also referred to as *neural implicit representations*³², utilize deep learning to encode spatial information within the weights of neural networks¹². These methods map sensory inputs onto continuous, high-dimensional latent spaces, contrasting with traditional methods that explicitly store occupancy or semantic attributes at discrete locations. Several distinct network architectures have been proposed:

- Neural Radiance Fields (NeRF)¹³ employ multi-layer perceptrons (MLPs) with positional encodings to map 3D coordinates and viewing directions into volumetric color and density estimates. This enables continuous and differentiable scene representations, primarily for view synthesis and rendering tasks.
- Occupancy Networks¹⁴ similarly rely on MLPs but focus on learning implicit occupancy functions. Given a 3D point, the network outputs a probability of occupancy, effectively defining decision boundaries that describe object geometry at arbitrary resolutions.
- Convolutional neural networks (CNNs)¹⁶ are typically used for grid-based inputs such as LIDAR or camera data. These networks exploit local spatial correlations through convolutional layers, enabling efficient feature extraction for tasks like semantic segmentation or motion prediction. Shallow CNN architectures have also been leveraged where they aim to make the learned kernel human interpretable. Despite this interpretability, they still require domain specific pretraining or fine-tuning².
- Recurrent Neural Networks (RNNs)¹⁵ incorporate temporal memory cells such as long short-term memory (LSTM) or gated recurrent units (GRUs) to create static scenes from sequential sensor observations.

While neural methods demonstrate high performance, their reliance on implicit representations presents challenges for interpretability, and safety validation, particularly in mission-critical systems². Furthermore, their architectures require retraining or fine-tuning for deployment across different domains or environments.

Vector Symbolic Probabilistic Computation

We developed VSA-OGM based on Spatial Semantic Pointers (SSPs)²⁴, which provide efficient probabilistic representations and real-valued encoding that can be leveraged to address the limitations of traditional

Table 10 | The mathematical symbols used to describe VSA-OGM

Symbol	Meaning
$\otimes$	circular convolution (binding)
$+$	element-wise addition (bundling)
$*$	Dot product
$\odot$	Hadamard product
d	vector dimensionality
n	environmental dimensionality
ϕ	Spatial Semantic Pointer (SSP)
ϕ_{axis}	axis SSP
$\phi_{axis}(x)$	$x \in \mathcal{R}$ encoded as a SSP
ϕ_{loc}	an n-d location encoded as an SSP
$\mathcal{F}$	Discrete Fourier Transform (DFT)
$\mathcal{F}^{-1}$	inverse DFT
$\ \bullet\ $	vector magnitude
l	length scale
λ	point cloud
j	number of points in each point cloud
t	time step
γ	n -dimensional coordinate vector
ψ	class vector
δ	number of tiles per dimension
ω	tile memory vector
β	singular tile
μ	center coordinate of β
r	disk filter radius
$M(x, y)$	disk filter mask for (x, y)
H	Shannon entropy
ρ	decision threshold
Ω	set of global memory vectors
N	the number of agents
Γ	a single agent
Pr	true probability value
$\mathcal{P}$	quasi-probability value
X	set of discrete samples along x axis
Y	set of discrete samples along y axis
$d(x, y, x', y')$	L2 distance between (x, y) and (x', y')
$OGM(x, y)$	probabilistic signed difference at (x, y)
$class(x, y)$	inferred class of (x, y) with ρ

and neural methods. SSPs are defined as d -dimensional real-valued hypervectors normalized to unit length. The mathematical symbols used in this manuscript are defined in Table 10. All fundamental VSA operations are compatible with SSPs where grouping multiple vectors together, bundling, is achieved through element-wise addition ($+$):

$$\phi(C) = \phi(A) + \phi(B). \quad (1)$$

Binding uses circular convolution ($\otimes$) to associate vectors together, as shown in Equation (2),

$$\phi(A) \otimes \phi(B) = \mathcal{F}^{-1}\{\mathcal{F}\{\phi(A)\} \odot \mathcal{F}\{\phi(B)\}\}, \quad (2)$$

where $\mathcal{F}$ denotes the Fast Fourier Transform³⁸. The resulting vector is orthogonal or nearly orthogonal to both inputs, enabling distinct

compositional representations. This operation underlies the slot-filler representations commonly used in VSA systems²⁴.

While many VSA systems³⁹ represent discrete concepts using randomly sampled, mutually dissimilar vectors, they lack the ability to encode continuous spatial information. SSPs overcome this limitation by enabling fractional binding, which generalizes the standard binding operation to continuous domains. Specifically, SSPs allow real-valued spatial coordinates to be encoded by raising the Fourier transform of a base axis vector to a real-valued power^{27,40}:

$$\phi_{axis}(x) = \mathcal{F}^{-1}\{\mathcal{F}\{\phi_{axis}\}^x\}; x \in \mathbb{R}. \quad (3)$$

This fractional binding mechanism is central to VSA-OGM, enabling continuous representations in high-dimensional space.

Individual fractionally bound vectors can be compared through a dot product operation. SSPs have a unique computational property where this dot product approaches a sinc function as vector dimensionality d approaches infinity³⁵. If x encoded as $\phi_{axis}(x)$ and the estimate of x' encoded as $\phi_{axis}(x')$, the pseudo kernel density estimator $k(x, x')$ is calculated as: $k(x, x') = \phi_{axis}(x) * \phi_{axis}(x')$. The width of the induced kernel can be adjusted to specific values by modifying Equation (3) to

$$\phi_{axis}(x) = \mathcal{F}^{-1}\{\mathcal{F}\{\phi_{axis}\}^{(x/l)}\}; x \in \mathbb{R}, \quad (4)$$

where l is the length scale parameter and there is a positive correlation between l and the kernel width. This adjustment allows users to modulate the noise sensitivity of the estimator. Despite a lack of widespread use, sinc kernels have proven to be more accurate, in terms of mean integrated squared error, versus other kernels such as the Epanechnikov kernel⁴¹.

As shown in Fig. 6, the dot product of multiple orthogonal SSPs produces a sinc function over the spatial domain, effectively performing quasi-kernel density estimation. This process parallels methods such as random fourier features⁴² and vector function architectures (VFAs)⁴³, where high-dimensional encodings are used to facilitate efficient kernel-based inference. Closely related to SSPs are Fourier holographic reduced representations (FHRRs)²², another class of VSAs that perform operations in the complex domain to represent structured information in high-dimensional spaces.

Prior work has used both SSPs and FHRRs to construct dynamic models of place cells for real-time spatial localization^{25,44,45}. Although these approaches capture aspects of the localization component of Simultaneous Localization and Mapping (SLAM)⁴⁶, they operate on sparse, point-based recognition rather than constructing dense spatial maps over extended environments or long time horizons. These methods are demonstrated in small-scale environments or over short temporal sequences, rather than the long time horizons of partial observability and large spatial environments required for probabilistic OGM.

A complementary line of work applies VSAs and resonator networks to factorization⁴⁷, scene understanding⁴⁸, and visual odometry⁴⁹. These systems reconstruct a scene by resonating a hypervector toward a stored template, stabilizing multiple views but not accumulating new information over time. In contrast, OGM requires integrating sparse, time-varying observations into a continually expanding probabilistic map as the agent explores the environment.

Building Occupancy Grids

OGM systems sample multiple point clouds $\{\lambda_0, \lambda_1, \dots, \lambda_t\}$. For each timestep t , λ_t consists of a coordinate vector γ and a class vector ψ . Since we are focusing on 2D OGM, all points clouds contain multiple 2D points representing x and y on the local reference frame. Moreover, the class vectors contain a single binary attribute signifying if a cell is occupied or empty. To ensure fairness with the baseline methods, all point clouds are preprocessed to the global reference with the same simultaneous localization and mapping method described in ref. 4.

A visualization of a labeled point cloud is provided in Fig. 7a, where an agent samples the environment using a series of raycasts. Occupied points

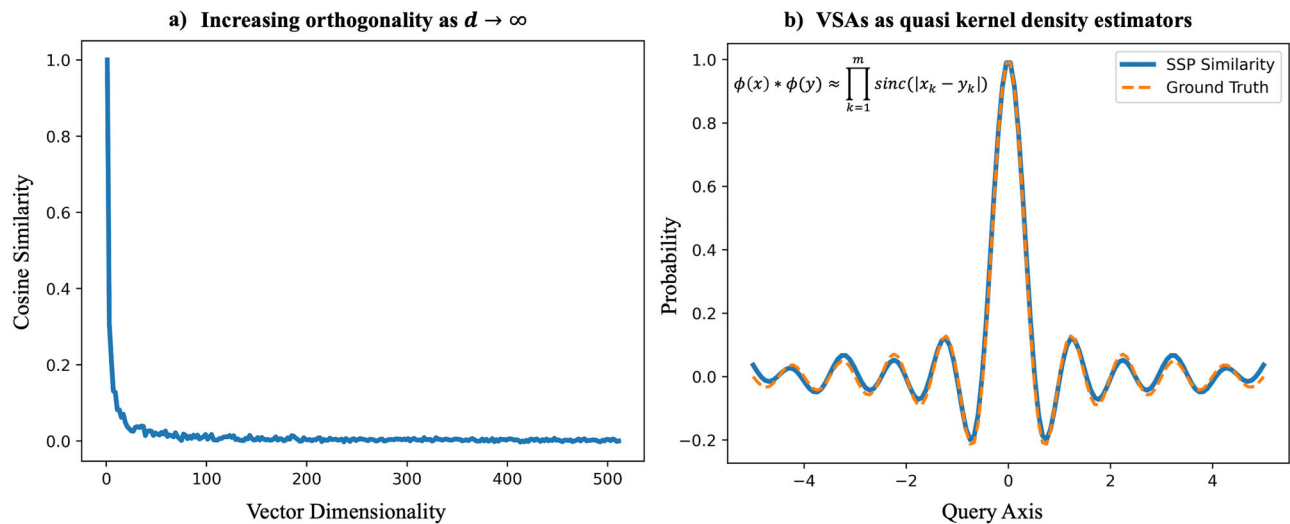


Fig. 6 | The properties of spatial semantic pointers (SSPs)²⁴. **a** The increasing orthogonality between pairs of randomly generated vectors as their dimensionality increases. **b** The dot product between two SSPs serving as a quasi kernel density estimator³⁵.

(green circles) are inferred from raycasts that terminate before reaching their maximum range, while points at the maximum range are labeled as empty (blue pentagons). Additional empty points are inferred by interpolating along the raycasts (blue circles).

Initialization

Multiple properties of the VSA system must be specified before any point clouds can be processed.

- **Vector Dimensionality d :** The size of the individual vectors utilized within VSA-OGM.
- **Length Scale l :** The width of the quasi-kernel induced by comparing multiple fractionally bound SSPs.
- **Axis Vector:** The basis of fractionally binding Cartesian information in VSA systems. In the case of 2D OGM, we define axis vectors ϕ_{axis}^x and ϕ_{axis}^y representing the x and y axes.

Environmental Discretization

Inspired by recent insights into chunking, where mammalian brains partition large scale spatial phenomena into distinct memory segments⁵⁰, we abstracted this principle to discretize the environment into a matrix of tiles. This tiling enables VSA-OGM to learn multiple regions of the environment independently, a critical capability given that the bundling operation assumes all stored vectors are correlated. Without this separation, representing the entire environment with a single vector would weight all spatial regions equally, disregarding the distance dependent structure of the space.

Specified by δ , we split each dimension into δ evenly sized and spaced tiles. To further discretize the environment, each tile is also separated on a class-by-class basis. As such, for a given experiment with δ tiles per dimension and 2 classes, there are $\delta^n \cdot 2$ memory vectors ω with d dimensionality:

$$\omega_1, \omega_2, \dots, \omega_{\delta^n \cdot 2}. \quad (5)$$

This process is demonstrated in Fig. 7a, where the environment is discretized into four tiles to create spatially distinct training regions. Observed points within each tile are aggregated and processed in parallel, facilitating structured and scalable downstream learning.

Encoding Point Clouds

Each point y_i in the point cloud λ_i is encoded into a set of location vectors $\phi_1, \phi_2, \dots, \phi_j$ by fractionally binding each coordinate of y_i along its respective dimension. This binding operation uses a variable length scale l and is

performed by expanding Equation (2).

$$\phi_i = \mathcal{F}^{-1} \left\{ \{ \phi_{axis}^x \}^{\gamma_i^x / l} \odot \mathcal{F} \{ \phi_{axis}^y \}^{\gamma_i^y / l} \right\}. \quad (6)$$

This process is repeated for all j points in λ_i resulting in a set of j location vectors representing each point in λ_i encoded into hyperdimensional space. Figure 7b visualizes this process, where occupied (green) and empty (blue) points are encoded by fractionally binding them with the axis basis vectors, producing the respective occupied and empty vectors for each observation.

Given a set of location vectors $\{\phi_1, \phi_2, \dots, \phi_j\}$ and their corresponding classes $\{\psi_1, \psi_2, \dots, \psi_j\}$ from λ_p , we follow Algorithm 1 to add the j points to the appropriate tile memory based on the closest tile and class. We begin by computing the center μ of each tile. Then, for each point, we identify the tile whose center is closest in Euclidean distance. Finally, we add the point's vector ϕ_i to the memory associated with the nearest tile and label ψ_i . Figure 7c visualizes this process, where occupied (green) and empty (blue) vectors are bundled into a single memory vector representing the cumulative representation of the observed point cloud. This process is also shown algorithmically in Algorithm 1.

Algorithm 1. Adding point vectors to tile memories

Require: $\{\omega_1, \omega_2, \dots, \omega_{\delta^n \cdot 2}\} \leftarrow$ tile memory vectors

Require: $\{\gamma_1, \gamma_2, \dots, \gamma_j\} \leftarrow$ points in Euclidean space

Require: $\{\phi_1, \phi_2, \dots, \phi_j\} \leftarrow$ location vectors

Require: $\{\psi_1, \psi_2, \dots, \psi_j\} \leftarrow$ class vectors

Require: $n \leftarrow$ environment dimensionality

Require: $\delta \leftarrow$ tiles per dimension

Require: $\psi_i \in \{0, 1\}$

Require: $n \geq 1$

Require: $\delta \geq 1$

Require: $j \geq 1$

$i \leftarrow 1$

$\{\mu_1, \dots, \mu_{\delta^n}\} \leftarrow \text{getTileCenters}()$

while $i \leq j$ **do**

$\beta \leftarrow \arg \min d_{L2}(\gamma_i, \mu_k)$

$\omega_{2\beta + \psi_i} \leftarrow \omega_{2\beta + \psi_i} + \phi_i$

$i \leftarrow i + 1$

end while

Decoding

For OGM, we focus on two key pieces of information: (1) which regions of the map are occupied, and (2) which regions are unoccupied or

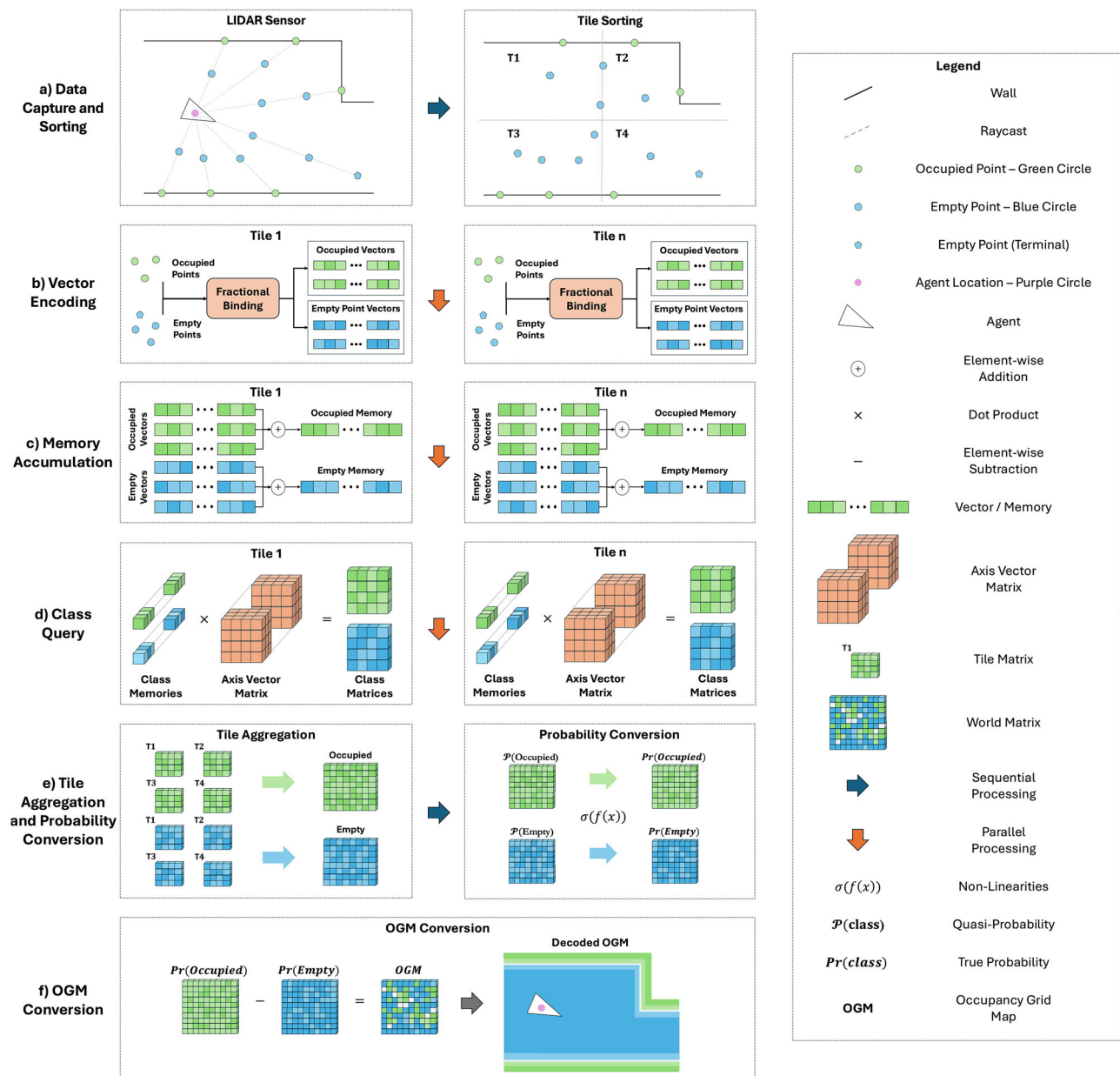


Fig. 7 | A high-level system overview of VSA-OGM. **a** Data capture and sorting with the training data being sampled from the LIDAR sensor before being sorted into individual tiles. **b** On a per-tile basis, the training data is fractionally bound into hyperdimensional vectors. **c** The vectors representing each data point are bundled into a single memory vector. **d** The aggregated class vectors are compared with the axis vector matrix, representing the tile's region encoded into hyperdimensional

space, resulting in a class matrix that contains quasi-probability values representing the likelihood of each class at each discretized point. **e** The individual class matrices from each tile are aggregated together before being post processed with a series of non-linear post processing steps to remove noise and extract encoded information. **f** The individual class matrices are combined into a single occupancy grid map with the probabilistic signed difference.

empty. Using the fully specified tile memory vectors $\omega_1, \omega_2, \dots, \omega_{\delta^n \times 2}$, we infer both occupancy and emptiness through deterministic vector operations combined with Shannon entropy⁵¹. This framework enables a probabilistic quantification of occupancy throughout the map. The decoding pipeline operates in four stages: (1) vector normalization, (2) similarity evaluation, (3) non-linear post-processing, and (4) entropy-based information extraction. We formally define and justify each of these steps in the following paragraphs. The overall process is illustrated in Fig. 7d–f. In Fig. 7d, individual memory vectors are normalized and compared against the axis vector matrix to resolve class similarity. These tile-level results are then aggregated in Fig. 7e, followed by non-linear post-processing to transform quasi-probabilities into valid

probability values. Finally, Fig. 7f shows the combination of class probabilities to produce the resulting OGM.

Vector Normalization

Throughout the continuous operation of any SSP-based VSA system, successive bundling will slowly increase the magnitude of the memory vectors. Therefore, a normalization operation transforming the memory vector back to unit length is required before querying any information:

$$\omega_i = \frac{\omega_i}{\|\omega_i\|}. \quad (7)$$

Similarity Evaluation

After vector normalization, we calculate the quasi-probability $\mathcal{P}$ of a specific location γ_i having a class of $\psi_i \in \{0, 1\}$ in a given tile β as:

$$\mathcal{P}(\psi_i|\gamma_i, \beta_i) = \omega_{2\beta+\psi_i} * \phi_{loc}, \quad (8)$$

where ϕ_{loc} is a location vector representing γ_i in hyperdimensional space as described in Eq. (6). The class of a region is calculated by expanding Eq. (8) across a matrix of ϕ_{loc} . Depending on the specific application and the amount of available memory, the computational complexity of calculating ϕ_{loc} can be minimized by precalculating a matrix of location vectors and storing them in memory. This process is shown visually in Fig. 7d, where both class memory vectors are multiplied with the axis vector matrix to produce a matrix of quasi-probability values representing the likelihood of each class.

Non-linear Post-processing

Even with vector normalization, the accumulation of noise from successive bundling renders direct decoding non-trivial. Many VSA frameworks address this challenge by employing clean-up memories or associative recall to project noisy vector representations onto more linearly separable factors^{40,44,48,49}. However, conventional codebook and resonator strategies are not applicable in the context of VSA-OGM. These approaches presuppose a fixed dictionary of factors and iteratively converge noisy vectors toward one of these predefined factors, after which an additional decoding step is required to obtain a real-valued location^{44,48,49}. By contrast, VSA-OGM requires well-defined similarity values across the entire spatial domain rather than the recovery of a single factor. Codebook- and resonator-based methods are therefore ill-suited to this task, whereas non-linear transformations applied directly to $\mathcal{P}$ yield more well-defined quasi-probability values.

We apply non-linearities to $\mathcal{P}(\psi_i|\gamma_i, \beta_i)$ such that they are normalized by the maximum quasi-probability of each class and squared:

$$\mathcal{P}(\psi_i|\gamma_i, \beta_i) = \left(\frac{\mathcal{P}(\psi_i|\gamma_i, \beta_i)}{\max(\mathcal{P}(\psi_i))} \right)^2. \quad (9)$$

We chose this combination of non-linearities based on the experimental analysis presented in *Results*. This combination of operations performs normalization based on the maximum observed quasi-probability value and forces all values to be between 0 and 1 with the Born rule from quantum probability theory⁵².

Algorithm 2. Calculating class-wise entropy

Require: $H(\psi) \leftarrow$ matrix of global entropy values
Require: $\mathcal{P}(\psi) \leftarrow$ quasi-probability map of class ψ
Require: $r(\psi) \leftarrow$ disk filter radius for class ψ
Require: $\psi \in \{0, 1\}$
Require: $x \in \mathbb{N}$
Require: $y \in \mathbb{N}$
 $x \leftarrow 1$
 $y \leftarrow 1$
while $x \leq \text{numCells}(x)$ **do**
 while $y \leq \text{numCells}(y)$ **do**
 $m_1 = M(x, y, r(\psi = 1))$
 $m_2 = M(x, y, r(\psi = 0))$
 $H(\psi = 1|x, y) = \text{entropy}(\mathcal{P}(\psi = 1), x, y, m_1)$
 $H(\psi = 0|x, y) = \text{entropy}(\mathcal{P}(\psi = 0), x, y, m_2)$
 $y = y + 1$
 end while
 $x = x + 1$
end while

Entropy Information Extraction

Shannon's theory of information has revolutionized modern communication systems and allows us to quantify information within noisy signals⁵¹. Recently, many works in material sciences⁵³ and radar imaging⁵⁴ use this well studied approach to extract feature rich information from noisy or otherwise incomprehensible data. Taking inspiration from these works, we employ an entropy-based mechanism to extract labeled spatial information from the class matrices calculated by applying Eqs. (8) and (9) across the entire map. This results in a matrix $\mathcal{P}(\psi)$ that can be conditioned on discrete coordinates within the Cartesian plane.

We apply localized Shannon entropy defined as:

$$H(\psi|x, y) = - \sum_{x,y} M(x, y) \mathcal{P}(\psi|x, y) \log \mathcal{P}(\psi|x, y), \quad (10)$$

where $x \in X$ and $y \in Y$, with X and Y denoting the sets of discrete coordinates along the horizontal and vertical axes of the environment. We define a binary mask function $M(x, y)$ such that it selects only the values within a radius r of a given cell (x, y) . The mask function is then defined as:

$$M(x, y) = \begin{cases} 1, & \text{if } d(x, y, x', y') \leq r \\ 0, & \text{otherwise} \end{cases}, \quad (11)$$

where $x' \in X$ and $y' \in Y$. Moreover, the Euclidean distance between the cell (x, y) and its neighbor (x', y') is defined as:

$$d(x, y, x', y') = \sqrt{(x' - x)^2 + (y' - y)^2}. \quad (12)$$

Using masks with radii $r(\psi = 1)$ and $r(\psi = 0)$ for both classes, this process is repeated across all discrete cells resulting in class-wise entropy values that increase class separability and reduce background noise. This process is shown programatically in Algorithm 2.

$$Pr(\psi_i|x, y) = \frac{e^{H(\psi_i|x, y)}}{\sum_{j=1}^2 e^{H(\psi_j|x, y)}} \quad (13)$$

As shown in Eq. (13), we apply the softmax function to convert the resulting class-wise entropy values to true probability values that sum to 1. This process is applied across all $x \in X$ and $y \in Y$.

$$OGM(x, y) = Pr(\psi = 1|x, y) - Pr(\psi = 0|x, y) \quad (14)$$

As shown in Eq. (14), we create our final map by applying the probabilistic signed difference to the individual class probabilities. This entire process is highlighted visually in Fig. 8, where the low fidelity quasi-probability values are post-processed into true probabilities with the combination of non-linearities, entropy analysis, and the softmax function. Lastly, the resulting OGM is created by using the probabilistic signed difference to aggregate the individual probability values into a single representation. A high-level visualization, within the context of the overall framework, is also shown in Fig. 7e, f where the quasi-probability class matrices are converted into true class probability matrices before being combined into a single OGM.

With the final OGM defined, we classify each point (x, y) as:

$$\text{class}(x, y) = \begin{cases} \text{occupied}, & OGM(x, y) \geq \rho \\ \text{empty}, & \text{else} \end{cases}, \quad (15)$$

where ρ is the decision threshold. The sensitivity of VSA-OGM to the proper specification of ρ is highlighted with the area under the receiver operating characteristic curve³⁰ and the negative log-likelihood in *Results*.

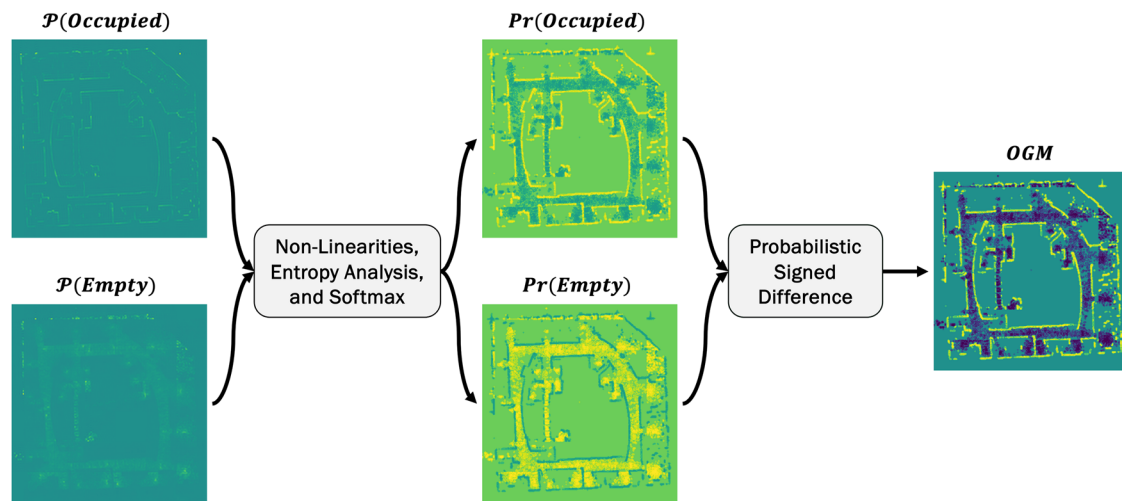


Fig. 8 | VSA-OGM's Decoding Process. Visualization of VSA-OGM's post-processing pipeline, which transforms quasi-probability values $\mathcal{P}$ for each class into calibrated probabilities Pr to generate the final occupancy grid map (OGM). The initial representation appears noisy or unclear because it directly reflects raw quasi-

probabilities. Applying VSA-OGM's non-linear decoding, entropy analysis, followed by softmax normalization and probabilistic signed difference, produces the clarified, class-separable dense OGMs.

Multi-Map Fusion

VSA-OGM supports multi-agent mapping through memory fusion. Given a set of N independently operating agents $\{\Gamma_1, \dots, \Gamma_N\}$, their tile memory vectors, the hyperdimensional vectors storing each agent's observations, can be merged if and only if three conditions are satisfied. First, each agent Γ_i must be operating on the same axis basis vectors for all dimensions of their operating environment. In this work focusing on two-dimensional OGM, all agents must have the same axis basis vectors ϕ_{axis}^x and ϕ_{axis}^y . Secondly, each agent should be learning the same number of classes with each having the same ψ value. Lastly, each agent must use the same number of tiles per dimension δ to separate their environment into distinct tiles along the global reference frame.

Algorithm 3. Fusing multiple agent's memory vectors

Require: $\{\Gamma_1, \dots, \Gamma_N\} \leftarrow$ the set of all agents

Require: $N \geq 2$

$\Omega \leftarrow \emptyset$

$i \leftarrow 1$

while $i \leq N$ **do**

$j \leftarrow 1$

while $j \leq \text{getNumMemories}(\Gamma_i)$ **do**

$\omega_j = \text{getMemory}(\Gamma_i, j)$

if $\omega_j \notin \Omega$ **then**

$\Omega = \Omega \cup \omega_j$ {Add ω_j to the set Ω }

else

$\Omega\{\omega_j\} = \Omega\{\omega_j\} + \omega_j$

end if

$j = j + 1$

end while

$i = i + 1$

end while

Assuming all three conditions are met, their unique tile memories can be merged into a cohesive set of global tile memories, representing the collective knowledge amongst the group. The process begins by initializing an empty set Ω that will contain the unique tile memory vectors within all agents. Now iterating over each agent, the following steps are followed. First, check if all unique tiles within the agent are represented within Ω . Second, if all tiles exist within Ω , proceed to the third step. Otherwise, an empty tile memory vector is initialized within Ω corresponding to the unique tile within the current agent. Third, use Eq. (1), to bundle each tile memory vector

within the current agent with the global tile memory vector within Ω . Lastly, repeat this process until all agents and all tile memories have been accounted for in Ω . This approach is also shown with pseudocode in Algorithm 3.

Data availability

The datasets that were used in each experiment are available here: <https://github.com/Parsa-Research-Laboratory/VSA-OGM>.

Code availability

The code for this study is available here: <https://github.com/Parsa-Research-Laboratory/VSA-OGM>.

Received: 30 September 2024; Accepted: 7 January 2026;

Published online: 02 March 2026

References

- Shi, L. & Luo, J. A framework of point cloud simplification based on voxel grid and its applications. *IEEE Sensors J.* **24**, 6349–6357 (2023).
- Wilson, J. et al. Convolutional Bayesian Kernel Inference for 3D Semantic Mapping, In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 8364–8370. <https://doi.org/10.1109/ICRA48891.2023.10161360> (London, United Kingdom, 2023).
- Thrun, S. Probabilistic robotics. *Commun. ACM* **45**, 52–57 (2002).
- Zhi, W., Ott, L., Senanayake, R. & Ramos, F. Continuous occupancy map fusion with fast bayesian hilbert maps. In *Proc. International Conference on Robotics and Automation (ICRA)* 4111–4117 (IEEE, 2019).
- Elfes, A. Using occupancy grids for mobile robot perception and navigation. *Computer* **22**, 46–57 (1989).
- Senanayake, R. & Ramos, F. Bayesian hilbert maps for dynamic continuous occupancy mapping. In *Proc. Conference on Robot Learning* 458–471 (MLR, 2017).
- Senanayake, R. & Ramos, F. Building continuous occupancy maps with moving robots. In *Proc. AAAI Conference on Artificial Intelligence* Vol. 32 (ACM, 2018).
- Duong, T., Yip, M. & Atanasov, N. Autonomous navigation in unknown environments with sparse bayesian kernel-based occupancy mapping. *IEEE Trans. Robot.* **38**, 3694–3712 (2022).
- Ghaffari Jadidi, M., Valls Miro, J. & Dissanayake, G. Gaussian processes autonomous mapping and exploration for range-sensing mobile robots. *Auton. Robots* **42**, 273–290 (2018).

10. Doherty, K., Wang, J. & Englot, B. Probabilistic map fusion for fast, incremental occupancy mapping with 3d hilbert maps. In *Proc. International Conference on Robotics and Automation (ICRA)* 1011–1018 (IEEE, 2016).
11. Zhai, P., Joseph, G., Myers, N. J., Onen, C. & Pandharipande, A. Spatial sparsity-aware radar-lidar fusion for occupancy grid mapping in automotive driving. *IEEE Sensors J.* (2025).
12. Van Kempen, R., Lampe, B., Woopen, T. & Eckstein, L. A simulation-based end-to-end learning framework for evidential occupancy grid mapping. In *Proc. Intelligent Vehicles Symposium (IV)* 934–939 (IEEE, 2021).
13. Tancik, M. et al. Block-nerf: Scalable large scene neural view synthesis. In *Proc. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 8238–8248 (IEEE Computer Society, 2022).
14. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S. & Geiger, A. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 4455–4465 (IEEE Computer Society, 2019).
15. Xiang, Y. & Fox, D. Da-rnn: Semantic mapping with data associated recurrent neural networks. Preprint at <https://doi.org/10.48550/arXiv.1703.0309> (2017).
16. Wu, P., Siheng, C. & Dimitris, N. M. Motionnet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. (2020).
17. van Kempen, R. et al. Data-driven occupancy grid mapping using synthetic and real-world data. In *Proc. International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)* 1–6 (IEEE, 2022).
18. Carlson, A. et al. Cloner: Camera-lidar fusion for occupancy grid-aided neural representations. *IEEE Robot. Autom. Lett.* **8**, 2812–2819 (2023).
19. Yang, S. et al. Lidar-llm: exploring the potential of large language models for 3d lidar understanding. *Proc. AAAI Conf. Artif. Intell.* **39**, 9247–9255 (2025).
20. Singhal, P., Walambe, R., Ramanna, S. & Kotecha, K. Domain adaptation: challenges, methods, datasets, and applications. *IEEE Access* **11**, 6973–7020 (2023).
21. Cheng, C.-H. et al. Neural networks for safety-critical applications - challenges, experiments and perspectives. In *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)* 1005–1006 (2018).
22. Plate, T. A. *Holographic Reduced Representation: Distributed Representation for Cognitive Structures* Vol. 150 (CSLI Publications Stanford, 2003).
23. Heddes, M., Nunes, I., Givargis, T., Nicolau, A. & Veidenbaum, A. Hyperdimensional computing: a framework for stochastic computation and symbolic ai. *J. Big Data* **11**, 145 (2024).
24. Eliasmith, C. *How to Build a Brain: A Neural Architecture for Biological Cognition* (OUP USA, 2013).
25. Dumont, N. S.-Y. *Symbols, Dynamics, and Maps: A Neurosymbolic Approach to Spatial Cognition*. Ph.D. thesis, University of Waterloo (2025).
26. Eliasmith, C. & Anderson, C. H. *Neural engineering: Computation, representation, and dynamics in neurobiological systems* (MIT Press, 2003).
27. Komer, B. *Biologically Inspired Spatial Representation*. Ph.D. thesis, University of Waterloo (2020). <http://hdl.handle.net/10012/16430>.
28. Gobin, D. et al. Exploration of novel neuromorphic methodologies for materials applications. In *PROC. International Conference on Neuromorphic Systems (ICONS)* 282–286 (IEEE, 2024).
29. Howard, A. & Roy, R. The robotics data set repository (radish) (2003). <http://radish.sourceforge.net/>.
30. Bradley, A. P. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognit.* **30**, 1145–1159 (1997).
31. Schreiber, M., Belagiannis, V., Gläser, C. & Dietmayer, K. A multi-task recurrent neural network for end-to-end dynamic occupancy grid mapping. In *Proc. IEEE Intelligent Vehicles Symposium (IV)* 315–322 (IEEE, 2022).
32. Zhang, H., Zou, Y., Yan, Z. & Cheng, H. Rapid-Mapping: LiDAR-Visual Implicit Neural Representations for Real-Time Dense Mapping. In *IEEE Robotics and Automation Letters*. Vol. 9, pp. 8154–8161 <https://doi.org/10.1109/LRA.2024.3440729> (2024).
33. Lang, A. H. et al. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. (2019).
34. Kleyko, D., Rachkovskij, D. A., Osipov, E. & Rahimi, A. A survey on hyperdimensional computing aka vector symbolic architectures, part I: models and data transformations. *ACM Comput. Surv.* **55**, <https://doi.org/10.1145/3538531> (2022).
35. Voelker, A. R. A short letter on the dot product between rotated fourier transforms. Preprint at <https://doi.org/10.48550/arXiv.2007.13462> (2020).
36. Jiang, P., Osteen, P., Wigness, M. & Saripalli, S. RELLIS-3D Dataset: Data, Benchmarks and Analysis. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 1110–1116. <https://doi.org/10.1109/ICRA48506.2021.9561251> (Xi'an, China, 2021).
37. Lee, G., Kim, S., Seo, D. & Kang, Y. Inverse reinforcement learning with dynamic occupancy grid map for urban local path planning: a CNN model approach. In *Proc. Intelligent Vehicles Symposium (IV)* 871–877 (IEEE, 2024).
38. Nussbaumer, H. J. *The Fast Fourier Transform* 80–111 (Springer Berlin Heidelberg, 1981).
39. Kleyko, D., Rachkovskij, D., Osipov, E. & Rahimi, A. A survey on hyperdimensional computing aka vector symbolic architectures, part ii: applications, cognitive models, and challenges. *ACM Comput. Surv.* **55**, 1–52 (2023).
40. Plate, T. A. Holographic reduced representations. *IEEE Trans. Neural Netw.* **6**, 623–641 (1995).
41. Tsybakov, A. B. Nonparametric estimators. In *Springer Series in Statistics* 1–76 (Springer New York, 2009).
42. Rahimi, A. & Recht, B. Random features for large-scale kernel machines. In *Proc. 21st International Conference on Neural Information Processing Systems (NIPS'07)*. 1177–1184 (Curran Associates Inc., Red Hook, NY, USA, 2007).
43. Frady, E. P., Kleyko, D., Kymn, C. J., Olshausen, B. A. & Sommer, F. T. Computing on functions using randomized vector representations (in brief). In *Proc. 2022 Annual Neuro-Inspired Computational Elements Conference* 115–122 (ACM, 2022).
44. Frady, E. P., Kanerva, P. & Sommer, F. T. A framework for linking computations and rhythm-based timing patterns in neural firing, such as phase precession in hippocampal place cells. In *Proc. Conference on Cognitive Computational Neuroscience* 1–5 (CNN, 2018).
45. Dumont, N. S.-Y., Orchard, J. & Eliasmith, C. A model of path integration that connects neural and symbolic representation. In *Proc. Annual Meeting of the Cognitive Science Society* Vol. 44 (eScholarship, 2022).
46. Durrant-Whyte, H. & Bailey, T. Simultaneous localization and mapping: part i. *IEEE Robot. Autom. Mag.* **13**, 99–110 (2006).
47. Karunaratne, K. G. et al. On the role of noise in factorizers for disentangling distributed representations. In *Annual Conference on Neural Information Processing Systems*. (2024).
48. Renner, A. et al. Neuromorphic visual scene understanding with resonator networks. *Nat. Mach. Intell.* **6**, 641–652 (2024).
49. Renner, A. et al. Visual odometry with neuromorphic resonator networks. *Nat. Mach. Intell.* **6**, 653–663 (2024).
50. Gobet, F. et al. Chunking mechanisms in human learning. *Trends Cogn. Sci.* **5**, 236–243 (2001).
51. Shannon, C. E. A mathematical theory of communication. *Bell Syst. Tech. J.* **27**, 379–423 (1948).

52. Born, M. Quantenmechanik der stoßvorgänge. *Z. Phys.* **38**, 803–827 (1926).
53. Liu, G. & Zheng, X. Fabric defect detection based on information entropy and frequency domain saliency. *Vis. Comput.* **37**, 515–528 (2021).
54. Chan, D., Gambini, J. & Frery, A. Speckle noise reduction in SAR images using information theory. In *Proc. Latin American GRSS & ISPRS Remote Sensing Conference (LAGIRS)* 456–461 (IEEE, 2020).

Acknowledgements

We acknowledge the technical and financial support of the Automotive Research Center (ARC) in accordance with Cooperative Agreement W56HZV-19-2-0001 U.S. Army DEVCOM Ground Vehicle Systems Center (GVSC) Warren, MI. DISTRIBUTION A. Approved for public release; distribution unlimited. OPSEC #8540.

Author contributions

S.S. developed the idea, performed design of experiments, analyzed the results, and wrote the manuscript. A.C. and D.G. helped with analysis and reviewing the manuscript. M.P. contributed to the idea, analysis, writing the manuscript, and supported/funded the project.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at <https://doi.org/10.1038/s44335-026-00052-w>.

Correspondence and requests for materials should be addressed to Shay Snyder or Maryam Parsa.

Reprints and permissions information is available at <http://www.nature.com/reprints>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2026