Bio-inspired monocular drone SLAM

OZAN ÇATAL, IDLab, Ghent University - imec, Belgium
TIM VERBELEN, IDLab, Ghent University - imec, Belgium
NI WANG, IDLab, Ghent University - imec, Belgium
MATTHIAS HARTMANN, imec, Belgium
BART DHOEDT, IDLab, Ghent University - imec, Belgium

Drone navigation in GPS-denied, indoor environments, is still a challenging problem. As drones can perceive the environment from a richer set of viewpoints, simultaneous localization and mapping (SLAM) becomes more complex, while having stringent compute and energy constraints. To tackle that problem, this research displays a biologically inspired deep-learning algorithm for monocular SLAM on a drone platform. We propose an unsupervised representation learning method that yields low-dimensional latent state descriptors, that mitigates the sensitivity to perceptual aliasing, and works on power-efficient, embedded hardware. We compare our method against ORB-SLAM3, and showcase increased robustness and an order of magnitude lower memory overhead.

CCS Concepts: • **Computer systems organization → Embedded systems**; Robotics.

Additional Key Words and Phrases: slam, drones, neural networks

ACM Reference Format:
Ozan Çatal, Tim Verbelen, Ni Wang, Matthias Hartmann, and Bart Dhoedt. 2022. Bio-inspired monocular drone SLAM. In *System Engineering for constrained embedded systems (DroneSE and RAPIDO '22), June 21, 2022, Budapest, Hungary*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3522784.3522788

1 INTRODUCTION

Autonomous indoor drone navigation is an open research problem applicable to many domains ranging from warehouse management to structural maintenance [12]. As opposed to mobile, driving platforms, drones operate in full 6 degrees of freedom and exhibit a more complex dynamics model. Moreover, drone platforms typically have more stringent constraints on size and power budget, which in turn limits the available resources on-board both in terms of memory and CPU cycles. An additional difficulty in autonomous flying in general is the lack of a clear odometry signal, as is often available in ground-based mobile robots.

Simultaneous localization and mapping (SLAM) covers a plethora of algorithms and techniques traditionally suited for mobile robot navigation [7]. Recently, algorithms fitting the SLAM framework are being adapted for drone platforms. Most attractive SLAM algorithms for indoor UAVs are the monocular SLAM approaches that use sparse feature extractors on the one hand, to accommodate the tight compute budget; and using some form of sensory odometry or sensor fusion scheme to accommodate the noisy odometry signal, such as ORB-SLAM [4, 10]. However, tracking sparse visual features limits the applicability in visually similar or textureless environments, as is the case for e.g. a warehouse setting.

DroneSE and RAPIDO '22, June 21, 2022, Budapest, Hungary

ACM ISBN 978-1-4503-9566-3/22/06.
https://doi.org/10.1145/3522784.3522788

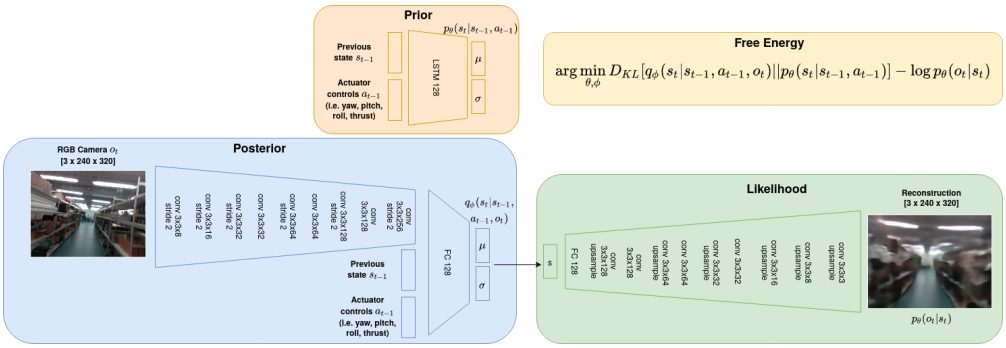


Fig. 1. Our generative model is composed out of three blocks. The Prior block needs to learn latent space dynamics, predicting the current state from previous state and actuator controls. The Posterior block combines the previous state and controls with features from a convolutional pipeline to make a posterior estimate of the current state. Finally, the likelihood model reconstructs the observation from state samples. These blocks are trained end-to-end by minimizing free energy, and at inference time only the Posterior network is used for inferring latent states.

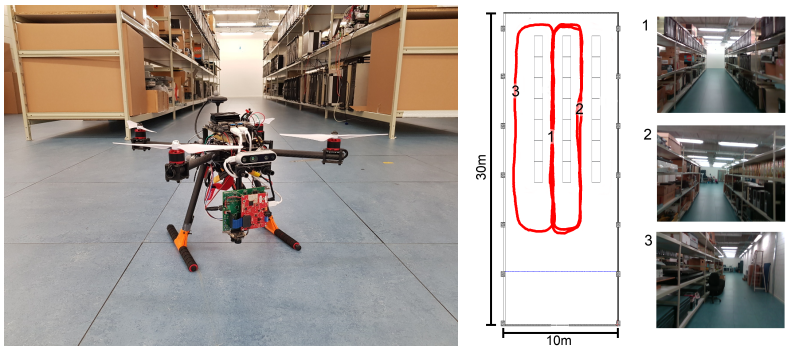
A different take on SLAM is a more bio-inspired approach that is based on recent findings on the inner workings of the hippocampal brain area in mammals. In this case, poses are represented as activations of grid and head direction cells in a continuous attractor network, and linked to representations of visual percepts [15]. The resulting map now becomes a graph with pose and visual representations, which is no longer a full metric representation, but rather a topological map.

In recent work [5] we presented LatentSLAM, which further improved upon these methods by using a learned latent representation of visual observations, adopting state of the art deep learning models. We train these latent representations in an unsupervised manner, by predicting future observations, in similar vein to how brains supposedly update themselves through predictive coding [17]. We illustrated how this yields more robust place descriptors that enable to mitigate perceptual aliasing, by evaluating on a mobile robot in a warehouse environment [5].

In this paper, we further extend LatentSLAM to work for drone navigation, in particular using only monocular camera images and a proprioceptive action signal, and assuring that our latent space model can be evaluated in real-time on an embedded compute platform. Furthermore, we compare the robustness and computational constraints of our approach to monocular ORB-SLAM. We present some initial results on a dataset collected with a real drone platform in a challenging indoor environment.

2 RELATED WORK

Simultaneous localization and mapping has been a long standing challenge in mobile robotics for decades [3]. This is even more demanding for aerial drones as they are naturally unstable, faster, have more degrees of freedom and are even more computationally constrained. Recently, a lot of research has focused in adapting various SLAM techniques for flying agents. These techniques often rely on the combination of geometric and semantic information. A key component of almost any SLAM algorithm is the tracking of the agent pose, often achieved through some form of dead reckoning in combination with proprioceptive measurements through an inertial measurement unit (IMU). IMU measurements are in most cases noisy, especially so for many commercial drone platforms.



(a) Drone platform (b) Test trajectory

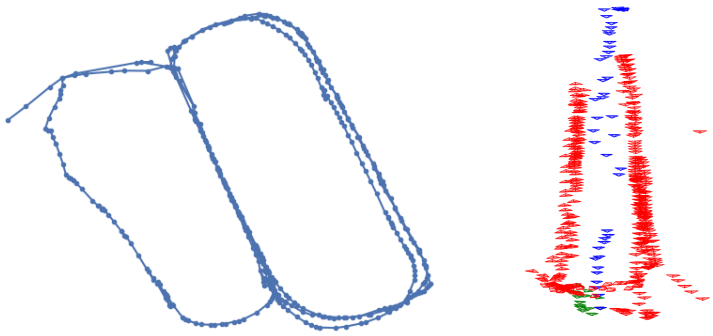
Fig. 2. Our experimental NXP drone platform, equipped with an Intel Realsense RGBD camera, a 79GHz TI radar and a Jetson Nano compute platform (a). In this paper, we focus on RGB camera data. We fly our drone in a warehouse-like lab environment, traversing similar looking aisles with racks and shelves (b).

IMU drift is often tackled by a form of sensor integration called visual-inertial odometry (VIO). For example, in [16] a localization method was proposed by integrating monocular SLAM and inertial measurement unit (IMU) data. Similarly, another work adopted sensor fusion of IMU data within ORB-SLAM for aerial navigation in GPS-denied environment [10]. A even more drastic approach to counter IMU drift, is to ignore the hardware IMU module and estimate the odometry signal straight from visual observations, also called visual odometry (VO). Air-SSLAM, a visual SLAM method which exploits a stereo camera configuration to detect keypoints and the correspondent feature descriptors over the pair of images [1]. Recently, VPS-SLAM proposes a lightweight and real-time visual semantic SLAM framework adapted to aerial robotic platforms by combining low-level VO along with geometrical information corresponding to planar surfaces extracted from detected semantic objects [2].

All these approaches have in common that they try to extract low-level features in a sparse way, in order to keep the computational load low enough for real time control. These approaches, however, typically fail to disambiguate similar looking percepts. This is typically the case in indoor environments such as warehouses, where the same features appear over and over again in different aisles. One approach is to build a dense metric map of features, typically a dense point cloud, but this comes at the cost of extra compute and memory. An other method is to build a topological map of the environment, as proposed by RatSLAM, a computational model based on the neural encoding of space by the rodent hippocampus, which enabled real-time SLAM on a two-wheeled robot [15]. Based on this, we recently proposed LatentSLAM, an extension of RatSLAM which uses a learnt, compact latent representation to disambiguate different locations [5]. In this work, we further investigate to what extent this approach can also be applied to drone navigation.

3 LATENT SLAM

LatentSLAM is a bio-inspired monocular SLAM algorithm loosely based on the rodent hippocampus. It builds a topological map of the environment solely based on visual cues and proprioceptive feedback. Instead of tracking sparse visual features, we encode camera observations into learnt low-dimensional latent state representations. At the same time, the proprioceptive signals are integrated using a pose continuous attractor network (CAN) [11]. Finally, the latent codes are



(a) LatentSLAM experience graph (b) ORBSLAM3 map

Fig. 3. Our LatentSLAM algorithm is able to recover a correct topological map of the traveled trajectory from monocular camera frames and proprioception (a). In contrast, ORBSLAM3 loses tracking at the ends of the aisle where there are little visual features and there is motion blur due to the drone turning. Axes are omitted since LatentSLAM produces topological maps, and the ORBSLAM map is only metric up to a scale factor.

combined with pose estimations into an experience graph, which represents a topological map of the environment.

To obtain a low-dimensional representation of camera observations, we train a generative model over observations o_t, actions a_t and hidden states s_t. We parameterize a posterior and likelihood model using deep neural networks, which are trained to maximize (a lower bound on) the log likelihood of observations $log(o_t)$, as done in variational auto-encoders (VAE) [13, 18]. In addition, instead of a fixed prior, our prior model is also parameterized as a recurrent neural network learning a prior over states given previous state and action. Hence, the latent space captures the dynamics of the environment up to a fixed time horizon by feeding action-observation-latent triplets to the encoder in a recurrent fashion as visualized in Figure 1. This model is trained end-to-end by maximizing the evidence lower bound (ELBO), or equivalently minimizing the free energy loss [8]:

$$L(\phi, \theta, \xi) = \sum_t D_{KL}\left[q_\phi(s_t|s_{t-1}, a_{t-1}, o_t) \;||\; p_\theta(s_t|s_{t-1}, a_{a-1})\right] - \log p_\xi(o_t|s_t). \tag{1}$$

Here q_ϕ denotes the posterior model, p_θ the prior model and p_ξ the likelihood model. At inference time, we only use the Posterior network to infer the current latent state of the system, given previous state, action and the latest camera observation.

4 EXPERIMENTS

To track the pose of the system, a pose CAN is parameterized as a 3D cube grid structure that wraps around the edges i.e., moving over the edge on one end moves the tracking to the opposite edge. The 3 axes of the cube represent the x, y pose and the θ orientation in the plane. Activation in a certain grid cell loosely represents the probability of the robot being in the corresponding pose. Grid cell activation is subject to attractor dynamics, where in steady state configuration the major activated cluster forms the current maximum likelihood estimate of the current robotic pose. At any given timestep, the estimates are shifted through the current proprioceptive signal and corrective activation from the current view cell estimate. Although a drone can in theory move along 6 degrees of freedom, we restrict ourselves in the pose CAN to x, y and θ, as we found these sufficient to track the drone in our indoor environments.

Distinct visited views are stored in a view cell array, where view cells link the latent code representing the obtained camera image together with the pose in which the view was encountered. This is similar to the concept of keyframes in feature-based SLAM method such as ORB-SLAM. When later a new observation matches a stored view cell, the corresponding pose gets increased activation in the pose CAN. This allows the agent to correct its own pose when encountering known locations. The linking of the pose to observations can be seen as a simplified form of Hebbian learning [6]. The injection of activation packets into the CAN from the view cell array is what enables the robot to relocalize itself in pose space, as subsequent observations in line with known correspondences will enforce the robot to adjust the current most likely pose estimate. View cell matching is achieved by taking the cosine similarity $\frac{s_{viewcell} \cdot s_t}{|s_{viewcell}||s_t|}$ and selecting the view cell with highest similarity, which yields a robust matching method. When no view is cell found within a certain matching threshold, a novel view cell is created and added to the array.

Finally, the experience graph embeds the interplay between the CAN and the view cell array in a global reference frame. This graph allows for pose estimation beyond the finite scope of the pose CAN. The experience nodes in the graph represent the view cell-pose pairs embedded in a non-metric space. A trajectory through the experience graph only loosely correlates to certain distance traversed (e.g. in meters) in the actual environment. The graph aims to capture the structure of the environment in terms of subsequent views, and uses the poses to give the map a meaningful visual ordering. Also, when view cells of different experience nodes are matched, this enforces loop closures by linking the corresponding experience nodes, and re-arranging the poses of the nodes in the graph using a graph relaxation procedure. More details on our approach can be found in [5].

To evaluate our LatentSLAM approach for indoor drone navigation, we conducted experiments in our industrial IoT lab environment, a 300 m^2 environment consisting of racks and shelves mimicking a warehouse. This is a challenging environment, as different aisles look very similar, making them hard to disambiguate from perception.

4.1 Setup

Our drone is an NXP Hovergames platform with a px4 flight-controller, mounted with an Intel Realsense RGBD camera and TI 79GHz mmWave radar as shown on Figure 2a. In this paper, we only use the RGB frames, but as future work we will consider the extra sensor modalities. We also equipped the drone with a Jetson Nano module as main compute resource.

4.2 LatentSLAM

We first collected data from 7 flights, totaling about 7500 frames, on which we train our model as described in Section 3. We train for 500 epochs using the Adam optimizer with initial learning rate 1e-4. After training, we use the Posterior model for estimating latent states from observations at inference time. We evaluate on a separate test flight, consisting of 1000 frames flying loops through three of the aisles as depicted in Figure 3b.

RGB frames are rescaled to a 320x240 resolution, and as source of proprioception we use the RC actuator control signals (yaw, pitch, roll and thrust) from the drone pilot. View cells are matched when the consine similarity is larger than 0.9.

4.3 ORBSLAM3

We compare LatentSLAM to the ORB-SLAM3 algorithm in VO mode as implemented by [4]. Here, RGB frames are provided in 640x480 resolution, with a maximum of 1000 features per image, and 8 levels in the scale pyramid.

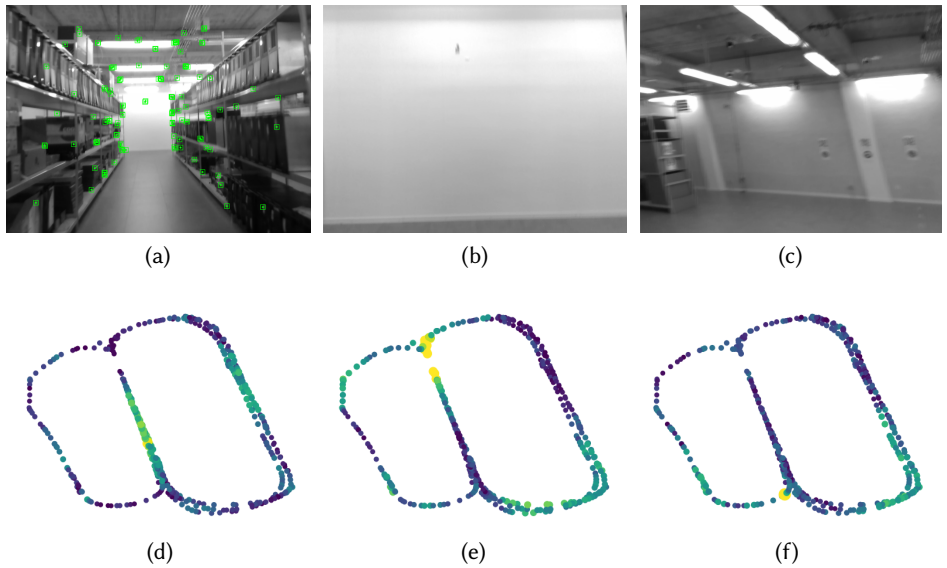


Fig. 4. Top: Three sample images from the test sequence where ORBSLAM tracking is good (a), or tracking is lost due to lack of features (b) or due to motion blur (c). Bottom: in all three cases LatentSLAM heatmaps yield sensible localizations based on the latent code, i.e. in the middle of the aisle (d), at the end of the aisle facing the white wall (e) or in the open space taking a turn towards the aisle (f).

4.4 Results

We first compare LatentSLAM with ORB-SLAM by inspecting both maps visually, as shown in Figure 3. Note that both maps are unit-less as LatentSLAM does not keep track of exact metric information, while ORB-SLAM in VO mode is not able to estimate the absolute scale of the map, rendering any recovered distances equally meaningless. The test sequence consists of a three passes through two hallways in the warehouse lab, which we are able to recover from the LatentSLAM map but not from the ORB-SLAM map. ORB-SLAM correctly recovers two aisles of the flight, but often loses tracking, resulting in different, separate pieces of the map, as depicted by the different colors.

Closer inspection reveals that ORB-SLAM loses tracking in cases where the camera image contains little texture or distinct features, or when the image is blurry due to the motion of the drone. This is especially the case at the far ends of the aisles, where either the drone is facing a white wall (Fig. 4b) or the drone is turning in the open space (Fig. 4c). LatentSLAM on the other hand, adopts a learnt latent descriptor for the whole image, and is able to correctly match the camera frames to locations in the experience graph. Figures (d-f) plots the corresponding LatentSLAM experience graphs, color mapped by cosine similarity with the latent code of the camera frame at hand, with view cell matches depicted in yellow.

4.5 Runtime

To enable real-time execution on a battery-powered drone platform, inference of our posterior model takes 25ms on a Jetson Nano board. Our latent space currently has 32 dimensions, which results in a low memory footprint for building and maintaining the map. The ORB-SLAM feature extraction pipeline is also known for its realtime performance. However, ORB-SLAM needs to keep

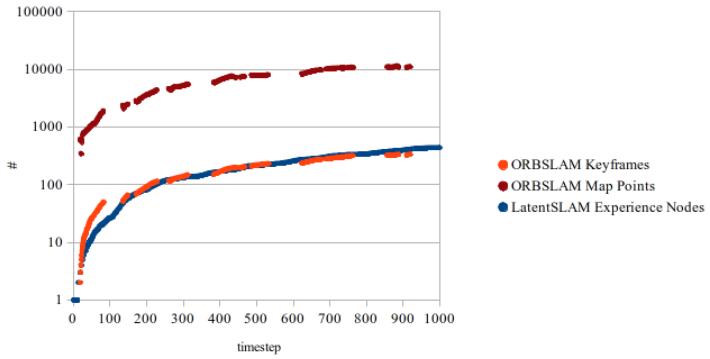


Fig. 5. Evolution of the number of ORBSLAM keyframes, map points and LatentSLAM experience nodes added to the map as the sequence progresses. Although LatentSLAM experience nodes scale similarly as ORBSLAM keyframes, ORBSLAM requires an order of magnitude more map points and corresponding feature descriptors in memory as the map expands. Missing data for ORBSLAM indicates periods where tracking was lost.

track of both a collection of map points with their feature descriptors, as well as a set of key frames from which these map points are viewed. In Figure 5 we see that the amount of ORB-SLAM key frames is similar to the number of experience nodes in the LatentSLAM graph, but the number of map points is an order of magnitude larger. Also note the missing datapoints in the ORB-SLAM graph, which are the periods that ORB-SLAM lost tracking.

5 DISCUSSION

It is clear that machine learning representations offer many advantages over hard coded feature descriptors or image representations. Learnt representations can typically achieve much higher compression rates, especially for high-dimensional sensory data [14], and exploit the various structures in the train distribution. The downside of course is that you need an initial training phase, either offline by collecting data upfront, either online when deploying the system. More importantly, the risk is that the system overfits to the training data, and does not respond well to changes in the environment. To what extent LatentSLAM can reuse its latent representations in other environments, or whether the representation can be fine-tuned online requires further research.

Researching bio-inspired approaches to SLAM might not only yield novel engineering solutions to the problem, but might also foster insights about how the brain might work. For example, combining representation learning by free energy minimization with hippocampal models for navigation might be compliant with predictive coding theories of the brain [8], and might spark novel process theories about navigation [19].

Finally, note that we still use a 3-DOF pose representation, which seems rather limited for a drone. However, we argue that for indoor environments explicitly modeling the height, pitch and roll does not add much information for correct mapping and loop closure. Although we did not fly at a fixed height during the test sequence, our system correctly matched corresponding locations. It might be interesting to investigate whether it modeled these extra degrees of freedom in the latent space. The simplicity from the 3-DOF pose representation is in part due to the fact that it is inspired by the way grid cells, place cells and head direction cells are found in the rodent

brain. The extension of this to flying mammals such as bats, however, is still a topic under active development [9].

6 CONCLUSIONS & FUTURE WORK

In this paper, we extended LatentSLAM, a bio-inspired SLAM method using unsupervised representation learning for aerial navigation in indoor environment. We demonstrated robust performance in mapping and navigation on a real-world drone platform using only monocular RGB and proprioceptive signals, while being able to execute on an embedded computation platform. As future work we plan to further extend and release our dataset with more sequences and more environments. We also plan to use the additional radar data to further increase the robustness, i.e. in low light or smoke conditions.

ACKNOWLEDGMENTS

O.Ç. is funded by a Ph.D. grant of the Flanders Research Foundation (FWO). This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" program.

REFERENCES

[1] P. Araújo, R. Miranda, D. Carmo, R. Alves, and L. Oliveira. 2017. Air-SSLAM: A Visual Stereo Indoor SLAM for Aerial Quadrotors. *IEEE Geoscience and Remote Sensing Letters* 14, 9 (2017), 1643–1647. https://doi.org/10.1109/LGRS.2017.2730883

[2] H. Bavle, P. De La Puente, J. P. How, and P. Campoy. 2020. VPS-SLAM: Visual Planar Semantic SLAM for Aerial Robotic Systems. *IEEE Access* 8 (2020), 60704–60718. https://doi.org/10.1109/ACCESS.2020.2983121

[3] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. 2016. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *Trans. Rob.* 32, 6 (Dec. 2016), 1309–1332. https://doi.org/10.1109/TRO.2016.2624754

[4] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. 2020. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM. *CoRR* abs/2007.11898 (2020). arXiv:2007.11898

[5] Ozan Çatal, Wouter Jansen, Tim Verbelen, Bart Dhoedt, and Jan Steckel. 2021. LatentSLAM: unsupervised multi-sensor representation learning for localization and mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*. https://doi.org/10.1109/ROBOT.2004.1307183

[6] Yoonsuck Choe. 2013. *Hebbian Learning*. Springer New York, New York, NY, 1–5. https://doi.org/10.1007/978-1-4614-7320-6_672-1

[7] Maksim Filipenko and Ilya Afanasyev. 2018. Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment. In *2018 International Conference on Intelligent Systems (IS)*. 400–407. https://doi.org/10.1109/IS.2018.8710464

[8] Karl Friston, Thomas FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, John O'Doherty, and Giovanni Pezzulo. 2016. Active inference and learning. *Neuroscience & Biobehavioral Reviews* 68 (2016), 862 – 879. https://doi.org/10.1016/j.neubiorev.2016.06.022

[9] Gily Ginosar, Johnatan Aljadeff, Yoram Burak, Haim Sompolinsky, Liora Las, and Nachum Ulanovsky. 2021. Locally ordered representation of 3D space in the entorhinal cortex. *Nature* 596, 7872 (01 Aug 2021), 404–409. https://doi.org/10.1038/s41586-021-03783-x

[10] S. J. Haddadi and E. B. Castelan. 2018. Visual-Inertial Fusion for Indoor Autonomous Navigation of a Quadrotor Using ORB-SLAM. In *Latin American Robotic Symposium*. https://doi.org/10.1109/LARS/SBR/WRE.2018.00028

[11] Dieter Jaeger and Ranu Jung (Eds.). 2015. *Attractor Neural Network*. Springer New York, New York, NY, 209–209. https://doi.org/10.1007/978-1-4614-6675-8_100038

[12] Yohanes Khosiawan and Izabela Nielsen. 2016. A system of UAV application in indoor environment. *Production & Manufacturing Research* 4, 1 (2016), 2–22. https://doi.org/10.1080/21693277.2016.1195304

[13] Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *CoRR* abs/1312.6114 (2013).

[14] Friso H. Kingma, Pieter Abbeel, and Jonathan Ho. 2019. Bit-Swap: Recursive Bits-Back Coding for Lossless Compression with Hierarchical Latent Variables. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). 3408–3417.

[15] M. J. Milford, G. F. Wyeth, and D. Prasser. 2004. RatSLAM: a hippocampal model for simultaneous localization and mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, Vol. 1. https://doi.org/10.1109/ROBOT.2004.1307183

[16] H. D. K. Motlagh, F. Lotfi, H. D. Taghirad, and S. B. Germi. 2019. Position Estimation for Drones based on Visual SLAM and IMU in GPS-denied Environment. In *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*. https://doi.org/10.1109/ICRoM48714.2019.9071826

[17] RP Rao and DH Ballard. 1999. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience* 2, 1 (January 1999), 79—87. https://doi.org/10.1038/4580

[18] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, Vol. 32.

[19] Ozan Çatal, Tim Verbelen, Toon Van de Maele, Bart Dhoedt, and Adam Safron. 2021. Robot navigation as hierarchical active inference. *Neural Networks* 142 (2021), 192–204. https://doi.org/10.1016/j.neunet.2021.05.010