In: *Proceedings of the 31st International Conference on Artificial Neural Networks (ICANN 2022)*, LNCS, Vol 13531, pp. 443-455, Bristol, UK, Sep 6–9, 2022

NeoSLAM: Neural Object SLAM for Loop Closure and Navigation

Younès Raoui[1], Cornelius Weber[2], and Stefan Wermter[2]

[1] LIMIARF Team , Department of Physics, Faculty of Sciences, Mohammed V University in Rabat, 4 Avenue Ibn Battouta, BP 1014, Morocco
www.fsr.ac.ma
y.raoui@um5r.ac.ma

[2] Knowledge Technology, Department of Informatics, University of Hamburg, Germany
www.knowledge-technology.info
{cornelius.weber,stefan.wermter}@uni-hamburg.de

Abstract. Simultaneous Localization and Mapping (SLAM) with fixed landmark objects creates topological maps by extracting semantic information from the environment. In this paper, we propose a new method for mapping, Neural Object SLAM (NeoSLAM), which uses objects seen in stereo images to learn associations between the pose of the robot and the observed landmark objects. We perform mapping with a biologically inspired approach based on creating patterns memorizing places in a network of grid cells and head direction cells. Our model is inspired by the object vector cells discovered recently by neuroscientists exploring the navigation of mammals. We model the firing field of these cells with a feed-forward neural network and create keyframes of objects with their 3D pose in a world-centered frame of reference. We train a Hebbian network connecting keyframe templates to the grid cells to memorize familiar places. We use the NeuroSLAM algorithm to train the grid cells and the head direction cells with the 4 Degree of Freedom (DoF) poses of the robot. Then, we detect loops in the trajectory by matching objects in the keyframes. Finally, we create an object experience map and correct the cumulative error if we detect loop closure candidates. Thus, our system performs object-based place recognition with a brain-inspired approach and produces 2D/3D object topological maps.

Keywords: SLAM · autonomous robotic · loop closure detection · fixed landmark objects · object vector cells · object experience map · neuroSLAM

1 Introduction

Robot Simultaneous Localization and Mapping (SLAM) creates a topological or metric map of the environment and simultaneously computes the robot's position within the map. New techniques exploit recent advances in neuroscience for making updated algorithms for robotic navigation. They map the environment with Continuous Attractor Networks [15], with Slow Feature Analysis [16] or

with Long Short Term Memory (LSTM) [3]. Indeed, they are inspired by the models of navigation of mammals using place cells, grid cells, and head direction cells. The place cells create a map of places, the grid cells compute the pose in the world frame, and the head direction cells are responsible for learning a model that predicts the angle of the head. In the current methods of brain-inspired navigation, several artificial neural network models are applied, such as Growing When Required Networks GWR [16], or deep learning methods such as autoencoders [9]. Teams of researchers learn how to compute the robot's pose with a dynamic neural field or a deep reinforcement network and predict the direction of the head of the robot with a Continuous Attractor Network (CAN) [3]. We create a brain-inspired topological map of places connected with edges representing the distance between them. We look for loop closure candidates by detecting familiar places to correct the pose of the graph's nodes. Recent SLAM techniques use visual objects like cars, trees, bicycles (outdoor scenes), chairs, desks, or laptops (indoor scenes) to represent maps.

Although these works revealed the importance of neural models in navigation, it is not clear how to use neural models for mapping with visual objects. New cells, named object vector cells, were discovered to be responsible for coding the objects in the entorhinal cortex in mouse cognition [8]. These object vector cells encode the vector between the mouse and the object. These cells spike when a mouse is near an object where the firing field has an elliptical shape. The current models train a neural network of grid cells and place cells with the appearance of images. Still, they do not train with objects, so they do not show a sensitivity to the high-level understanding of a given scene. Even so, the complexity of training only with visual landmarks is high, especially in large-scale environments.

This paper develops a new navigation system named NeoSLAM (Neural Object SLAM). We model the object vector cells using a feed-forward network to predict the firing rate from the direction and the distance from the object to the robot. Then we train a three dimensional CAN to model the grid cells. We detect loop closure candidates using the cluster spectrum correspondence method to match objects between two places [5]. We apply a Hebbian rule to learn a model that predicts the familiarity of a place, and we create an object experience map to avoid the problem of multiple representations of the environment and the hash collisions. Thus, we propose a new architecture of SLAM and we found that the translational error of the mapping is reduced when we relax the object experience map after detecting a loop. The score of matching keyframes is maximal when a familiar place is seen. We start with presenting the module-based architecture of our system NeoSLAM; then give the results, and discuss the implications.

2 Related works

In many works, SLAM considers objects as visual landmarks because they have semantic labels. They improve the data association with techniques such as particle filters [4] which maximizes the likelihood of the estimation of the map with the semantic labels of the observed objects. Also, they construct topological maps with methods of image segmentation based on Convolutional Neural Net-

works [1] or with algorithms of object detection with geometrical primitives [5]. Loop closure with objects was developed in many algorithms, e.g. [4,5], because it reduces the error of the pose and promotes stability in the CAN. Also, new methods for the visual control of autonomous systems for short or end-to-end driving have been developed with variational neural networks or continuous-time neural networks [11].

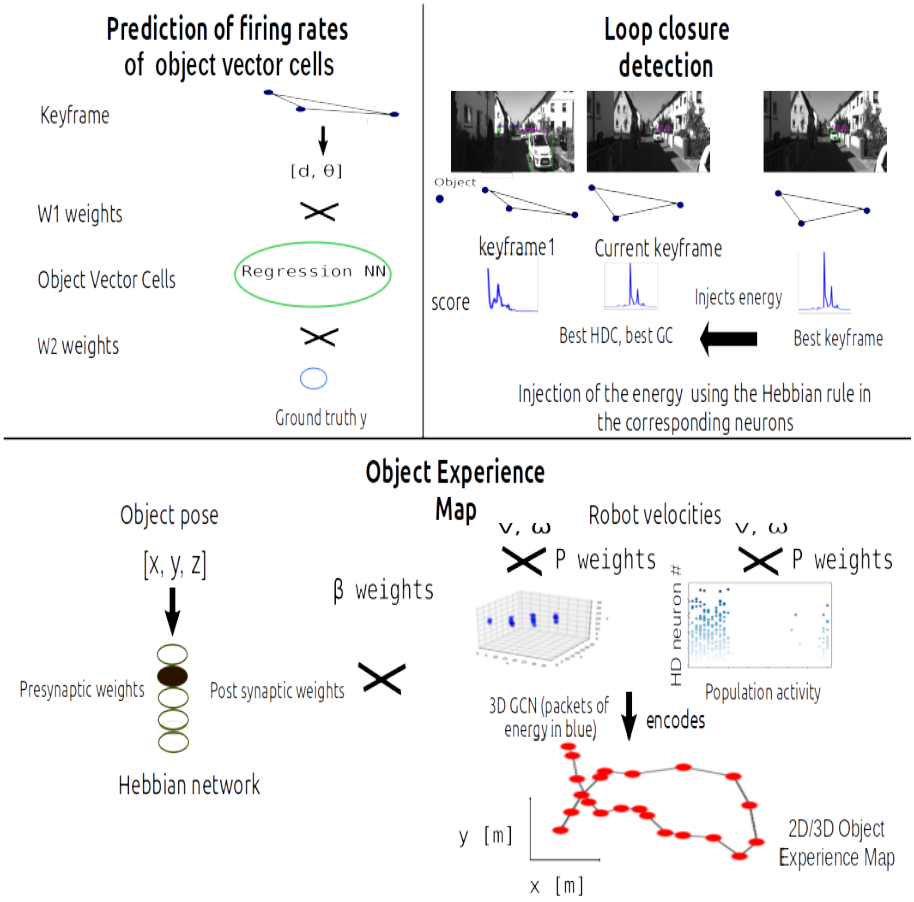


Fig. 1: The Object Vector Cells (ovc) model that learns their activity by training a regression neural network : inputs are the distance and the direction from the robot to the object of the keyframe, the output is the normalized sum of the firing rate of the active ovc (top left). The loop closure scheme based on the computation of the optimal score of matching objects [5](top right). The model of NeoSLAM that trains a GCN and HDC for the path integration with the velocity inputs. It trains a Hebbian network for simulating the associative memory of places(bottom)

Several works in modern robotics have implemented the model of grid cells and place cells by creating an associative memory of places with the Growing-When-Required networks [16]. In addition, grid cells have been modeled with LSTM for path integration or goal-oriented navigation implemented with deep reinforcement learning [3]. This shows us that neural networks can be used in robot navigation and place recognition. Furthermore, in bio-inspired navigation, the RatSLAM and NeuroSLAM are neural systems that create an associative memory for navigation through a 2D/3D CAN to obtain patterns of the robot pose and to make associations with the perceived images via Hebbian rules. They also creates a 2D/3D experience map to visualize the internal activity of the 2D/3D CAN [2,12,15]. Finally, as discovered in [8], the cells named object vector cells in the entorhinal cortex are responsible for mammals' navigation with objects using a vector representation of the pose of the objects necessary for spatial memory. This gives us the opportunity to make a neural model of object mapping for robots.

3 Description of the NeoSLAM system

The NeoSLAM system (see Fig. 3a) is initialized with the starting pose state of the robot and the objects (see Fig. 2a) detected with the Yolo software. A feed-forward neural network for the regression model named Object Vector Cells Network (OVCN) was developed to model the object vector fields of the object vector cells. We use the predicted responses of the OVCN to learn neural associations between the objects and the robot's pose when a familiar place is perceived. Then, we create keyframes, which are assembled to make an object experience map from the patterns of the 3D CAN, and correct the map using the robot's pose.

We create our system following the steps described below, from initializing the frame with objects and the robot pose to correcting the map:

1. We detect objects and describe them with ORB features; then, we create the frames at each step using the robot's pose, quaternion, and translational and angular velocities.

2. We create the keyframes that contain the current object's vertex and all object vertices within a threshold distance.

3. We train a regression feed-forward neural network to model the firing rate response of the object vector cells.

4. We train grid cells and head direction cells using the NeuroSLAM technique [15].

5. We detect loop closure candidates by matching the keyframes of objects [10]; we make an object experience map with the created keyframes, then we correct the map using the relaxation algorithm of the graph of the chosen keyframes.

3.1 Object Detection and Description for Scene Understanding with Appearance

The frame is created by computing the 3D pose of the robot, the quaternion of the orientation, and the translational and rotational velocities. Objects are detected from each new frame with Yolo [13], which provides the bounding boxes, the labels, and the 3D poses of the objects in the camera and the absolute frames. The depth of the objects is computed by matching and triangulating the ORB feature points from stereo images provided in Kitti datasets with the calibration parameters. We extract the objects with Yolo in the stereo images, match them, then compute the 3D pose in the robot coordinate frame – we compute the 3D pose in the world frame using the odometry data, which is affected by drifts. Then, we represent an object with the pixel coordinates of the upper left and the lower right corners, the label, the identifier, and the current left image.

3.2 Keyframe detection and Loop Closure detection

We first create the keyframe composed from the current object and all objects within a distance $e^{max} = 1.5m$ away.

The distance between two objects is given as:

$$e_{ij} = (||cv_j - cv_i||, \Sigma_{2x2}) \tag{1}$$

where i, j are the indices of the two objects. cv_i and cv_j are the 3D coordinates of their centers in the absolute frame. Σ_{2x2} is the covariance error on the distance between two objects (we set it as the identity matrix).

Keyframes matching

Given G that represents the set of all detected objects, and the current keyframe's objects G_1, we define $\{G_2\} = G \setminus G_1$ to not compare a keyframe to itself. G_1 is composed of objects represented with pairs of objects $(i, j)_k$ for $k = 1...n$. $\{G_2\}$ is composed of objects $(i^{'}, j^{'})_k$ for $k = 1...m$. (n and m are the numbers of objects in the two datasets). We perform a one-to-many mapping from G_1 to $\{G_2\}$ [10].

We calculate the affinity M with the following method:

$$H^r_{iji'j'} = e^{max} - ||e_{ij} - e_{i'j'}|| \tag{2}$$

$$H^l_{iji'j'} = \begin{cases} 1 & \text{if } label_i \text{ matches } label_j \text{ and } label_{i'} \text{ matches } label_{j'} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

and

$$M_{iji'j'} = \begin{cases} \mathrm{H}^r_{iji'j'} \cdot H^l_{iji'j'} & \text{if } i \text{ matches } j \text{ and } i' \text{ matches } j' \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

We measure the similarity between two objects by comparing the images where they appear using the Root Mean Square Deviation (RMSE = 80).

The best matching X^* is given by finding the optimal solution of this equation. X is the assignment matrix from G_1 to G_2:

$$X^* = argmax\ X^T M X \tag{5}$$

3.3 Training the Grid Cell (GCN) and the Head Direction Cell Networks (HDN)

We train the GCN and HDC to create a memory of the poses of the robot that move to stable states in the attractor neural network. In the NeuroSLAM, we initialize the GCN and HDC with a first pattern, then compute the next pattern by doing path integration based on velocities computed with visual odometry. We calculate the similarity between patterns by computing the distance between the poses encoded in the cells (x,y,z, and θ) directions [15].

The weights of the GCN are given as follows:

$$\epsilon^{gc}_{u,v,w} = \frac{1}{\delta_x\sqrt{2\pi}} \cdot e^{\frac{-u^2}{2\delta_x^2}} \cdot \frac{1}{\delta_y\sqrt{2\pi}} \cdot e^{\frac{-v^2}{2\delta_y^2}} \cdot \frac{1}{\delta_z\sqrt{2\pi}} \cdot e^{\frac{-w^2}{2\delta_z^2}} \tag{6}$$

where (u, v, w) represents the estimated distance between two robot poses in the absolute space, and δ_x, δ_y and δ_z are constants of variance for the 3D spatial distribution.

For modeling the HDC, we create a 1D continuous attractor network using the yaw Euler angle and 36 neurons to represent a single pattern of this angle.

3.4 Modeling the firing fields of the Object Vector Cells

Overview of the Object Vector Cells Object vector cells are mapping neurons located in the medial entorhinal cortex in mammal brains [8]. They are responsible for the navigation of mice, and react to objects of different sizes and types. They fire when the mouse is at a certain distance from particular confined objects [8]. The authors suggest that object vector cells intermingle with the GCN and HDN; their function can be replicated using a feed-forward network and a Hebbian network. We represent the vector information with the distance and the direction to the object (see Fig. 2b). Figure 2c shows a firing map of object vector cells 2 and 9.

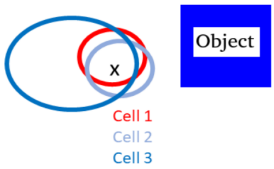
Ground truth of firing field of the object vector cells Let $d(x, y)$ be the distance between the robot pose and an object pose and let ϕ be the direction between the object and the robot in a world-centered reference frame. We suppose for simplicity that each object activates only one object vector cell. The experiments were performed on a mouse in [8]. The firing rate of the OVCN is modeled with an elliptical field which suggests that the firing rate will decrease from its defined maximum, relative to the preferred vector between the mouse in a cage and the object.

$$f(d) = a \cdot \exp[-d^T A\, d] + b \tag{7}$$

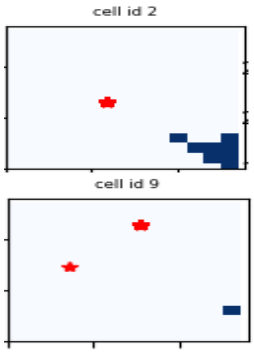
We set $a = 1.2$ Hz and $b = 0$. A is a diagonal matrix and R is a rotation matrix:

$$A = R^T A R \tag{8}$$

(a) Detection of object "car" with Yolo (Kitti dataset) (the car represents the rat in NeoSLAM)



(b) Illustration of 3 object vector cells activated when the agent (rat x) is near an object



(c) The object vector firing field of two cells (blue), and objects (red stars)

Fig. 2: Illustration of object detection and object vector cells

where

$$R = \begin{bmatrix} cos(\phi) & sin(\phi) \\ -sin(\phi) & cos(\phi) \end{bmatrix} \tag{9}$$

and

$$A = \begin{bmatrix} \frac{1}{2 \cdot \sigma^2} & 0 \\ 0 & \frac{1}{2 \cdot \sigma^2} \end{bmatrix} \tag{10}$$

The model We propose to train an Object Vector Cell Network with a feed-forward neural network for regression (OVCN). We consider the values of the model in [8] as the ground truth (see Eq. 7). The learning rate is 0.001, and the number of epochs is 10 (20 keyframes in each epoch). The inputs of the model are the distance and the direction of the object from the robot, and the output is computed using N hidden neurons. We update the OVCN with the back-propagation of the error in the neural network.

We have:

$$X_t = [d_t, \phi_t] \tag{11}$$

X_t is the observation at time t, and the ground truth of the firing rate of the activated object vector cells is computed with the function f (see eq. 7):

$$y_t = f(X_t)$$

The output of the network is:

$$O_t = g(W \cdot X_t + b) \tag{12}$$

where g is an exponential linear unit; W are the weights of the OVCN. O_t is the predicted value of the firing response of the OVCN.

We compute the loss of the training using L2 norm as loss function:

$$\mathcal{L}(y_t, O_t) = \frac{1}{2}(y_t - O_t)^2 \tag{13}$$

3.5 Learning associations between Keyframe templates and GCN

We create a single-layer neural network where inputs are the keyframes objects and the outputs are the neural excitation computed with the OVCN. Associations between the current pattern of the OVCN and GCN are learned using the Hebb rule between the pre-synaptic neurons of the keyframes, which fire simultaneously with the postsynaptic one of the GCN. The weight β between the two neurons is learned (see Fig. 1):

$$\beta^{t+1}_{k,x,y,z,\theta} = max(\tau O^k_t \cdot P_{x,y,z,\theta}, \beta^t_{k,x,y,z,\theta}) \tag{14}$$

where $P_{x,y,z,\theta}$ is the neural activity of the grid cell of coordinates (x, y, z), and its corresponding head direction cell firing at the angle θ. O^k_i is the activity of the template of keyframe k. It is given as $O^k_t = \frac{1}{N}\sum_{i=1}^{N} O^k_{ti}$, where N is the number of objects in the current keyframe.

3.6 Object Experience Map creation

The next step of our work is to create an experience map to overcome the problems of multiple scene representations and hash collisions. For this purpose, an Object experience map is built, inspired by the NeuroSLAM algorithm. We initialize the map with the first pose of the robot given with the following state:

$$Xr_t = \begin{bmatrix} x_t & y_t & z_t & q_t & v_t & \omega_t \end{bmatrix} \tag{15}$$

where x_t, y_t, z_t is the robot pose in the absolute frame. q_t is the quaternion of the orientation. v_t and ω_t are the translational and rotational velocities.

We represent an experience with the following vector:

$$exp_t = \begin{bmatrix} P^{gc}_t, P^{hdc}_t, Xr_t, KF_t \end{bmatrix} \tag{16}$$

where P^{gc}_t and P^{hdc}_t are the poses of the current active pose cell and the current active head direction cell of the highest energy. KF_t is the index of the current active keyframe template. We compute the similarity score:

$$S_t = \mu^{gc}|P^{gc}_i - P^{gc}_t| + \mu^{hdc}|P^{hdc}_i - P^{hdc}_t| \tag{17}$$

i is the index of the previous experiences, KF_t is equal to the index of the familiar keyframe if the score of matching is higher than a threshold; else, it is equal to zero. Also, another criterion for creating a new experience, proposed in NeuroSLAM, is that the similarity score S_t is higher than S_{max} for all the previous experiences. Concerning links, if we create a new experience, we link it to the last one by setting the euclidean distance value as a cost; if we find a similar experience, we relate it to the current one.

Processing loop closure candidates and Map relaxation We calculate the distance between the pose of the candidate keyframe in the grid cell network and the head direction cell network, and the current keyframe. If this distance is higher than a threshold (near seven bins in the GCN), we discard it. Otherwise, we excite the grid cell connected to the keyframe cell.

We correct the map by updating the poses of the current experience (t) and the connected experience $(t+1)$. We apply the following update equation where c_f is the factor of the correction [15]. Having $Xr_t = [x_t, y_t, z_t]$, the pose of the experience, and the pitch angle θ_t:

$$Xr_t = Xr_t + (Xr_{t+1} - l) \cdot c_f \tag{18}$$

where l represents the displacement from the experience t to the experience $t+1$.

4 Discussion and results

We used the Yolo software to experiment with object detection from extracted frames [13]. We use the scheme in the figure 1. We use viso2ros for the visual odometry [7]. We test our system with sequence 5 of the Kitti odometry dataset under ROS Noetic distribution [6] because it contains several loops and the ground truth data.

4.1 Training the OVCN

We can model the object vector fields with an OVCN that predicts the firing fields when a keyframe is created or recognized. We find that the loss of the training of the OVCN decreases with the number of iterations (see Fig. 4a). We could improve the model of object vector cells with a single layer Hopfield neural network HNN, which stores the pose of the objects, their size, color, and labels. This network will converge to a stable state in the neural field that memorizes places with their furniture (objects). Its inputs are the object features, and the weights are initialized with random values.

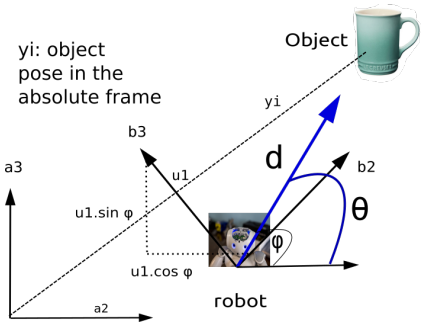
4.2 Loop closure detection

We create the frames, detect the objects, and triangulate 7 ORB keypoints of these objects in the right and the left image to compute the depth (baseline = 0.53m) [14]. Our method of matching gives good results, as shown in Fig. 4d. We find three candidate hypotheses of loop closures having a score higher than 1. Using the memory patterns in the GCN and HDN, we select the most accurate candidate identified by the highest score. Moreover, we improve the time complexity of the algorithms of G_2 creation by using a dynamic programming method. We matched in fig. 4c the initial keyframe to the stored keyframes, and we found that the loop closure is detected at the key position 200.

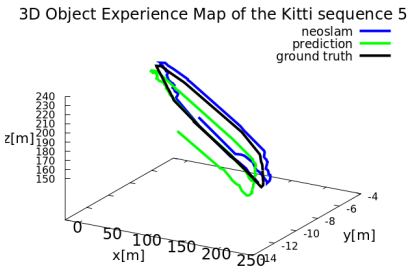
4.3 Object experience map creation and update

Figures 3d and 3b show in 2D and 3D, respectively, results for mapping using NeoSLAM compared to noisy data and ground truth. We apply a Gaussian noise to the ground truth between 0 and 1m/s for the translational velocity, 0 and 0.5 rad/s for the angular velocity to obtain the predicted experience map.

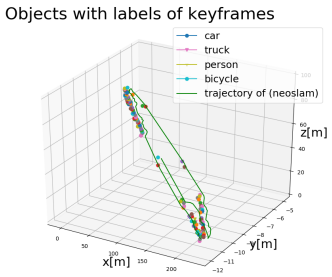
We find that the updated experience map is more accurate, as is proved by the calculation of the Euclidean distance in Fig. 4b. Figure 3c shows the different labels of objects that compose the keyframes created during the navigation (car, bicycle, person, and truck).



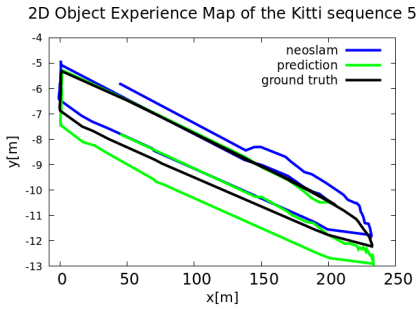
(a) The scheme of the environment



(b) The ground truth, predicted and corrected Object experience map in 3D



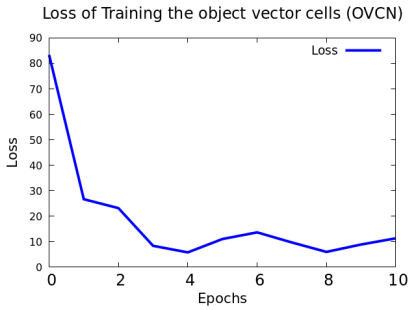
(c) The trajectory with labels of the objects of the keyframes



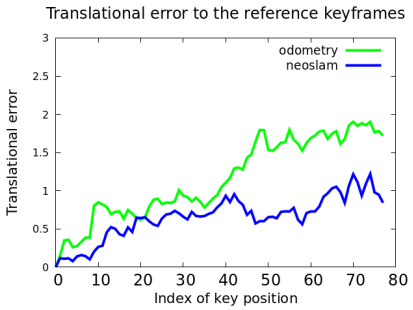
(d) The ground truth, predicted and corrected Object experience map in 2D

Fig. 3: Results of the mapping with NeoSLAM

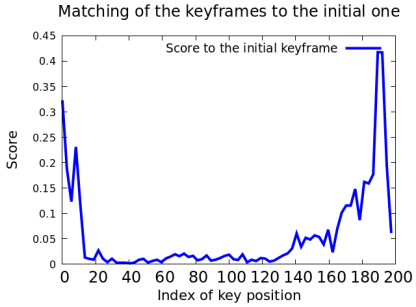
We showed that extracted objects from images could play a role in the navigation of robots, better than processing only images with their intensities. We remark that two parameters affect the precision of the correction: the distance between objects i and j, d_{ij} and the distance from the robot to the object i r_i. We notice that if d_{ij} is close to 10 meters and r_i is close to 1 meter, the correction is more accurate. Moreover, the correction accuracy is improved when we increase the dimension of the grid cells or the number of iterations for experience map correction or the correction rate.



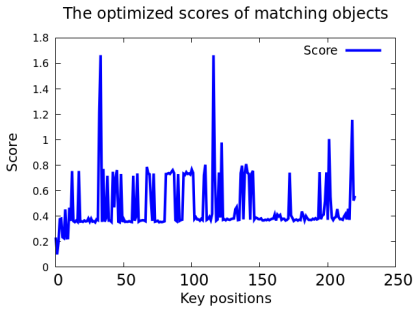
(a) The loss of training of the OVCN



(b) The translational error



(c) Matching keyframe 1, the loop closure is at the key position 190



(d) 3 candidates for loop closure (score>1),the correct candidate at key position 225

Fig. 4: Results of the OVCN modeling and the loop closure detection with NeoSLAM

5 Conclusion

This paper presents a new method of creating Object experience maps with visual objects. We propose to model the firing rate of the object vector cells with a OVCN neural network. Loop closures are detected by matching keyframes with their objects. We train a 3D CAN to create an associative memory of localization and mapping. A Hebbian neural network creates associations between the objects and the robot's pose. Our loop closure method enables robust creation of an object experience map of the robot. We find that the accuracy of the correction depends on the distance between objects and the depth of the objects, which is not the case for other methods of brain-inspired SLAM, that do not use single objects for anchoring. Thus, our system performs object-based place recognition with a brain-inspired approach. We propose in the future to model the object vector fields with a deep learning method such as LSTM in a dynamic environment. Next, we suggest interpreting visual objects in terms of their ORB visual words by learning a visual dictionary during the loop closure detection. However, NeoSLAM doesn't consider dynamic objects which is the case in many

robotic scenarios. Also, the time of computation in the loop closure detection still considerable even if we do dynamic programming. NeoSLAM could have several applications, such as exploring in depth the functions of the object vector cells to understand them better or doing SLAM in an indoor environment where the semantic information is important.

References

1. Ambruş, R., Bore, N., Folkesson, J., Jensfelt, P.: Meta-rooms: Building and maintaining long term spatial models in a dynamic world. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1854–1861 (2014). https://doi.org/10.1109/IROS.2014.6942806
2. Ball, D., Heath, S., Wiles, J., Wyeth, G., Corke, P., Milford, M.: OpenRatSLAM: an open source brain-based SLAM system. Autonomous Robots **34**, 149–176 (2013)
3. Banino, A., Barry, C.: Vector-based navigation using grid-like representations in artificial agents. Nature **557**, 429–433 (2018)
4. Doherty, K., Baxter, D., Schneeweiss, E., Leonard, J.: Probabilistic data association via mixture models for robust semantic SLAM. 2020 IEEE International Conference on Robotics and Automation (ICRA) pp. 1098–1104 (2020)
5. Finman, R., Paull, L., Leonard, J.: Toward object-based place recognition in dense RGB-D maps (2015)
6. Fritsch, J., Kuehnl, T., Geiger, A.: A new performance measure and evaluation benchmark for road detection algorithms. In: International Conference on Intelligent Transportation Systems (ITSC) (2013)
7. Geiger, A., Ziegler, J., Stiller, C.: Stereoscan: Dense 3d reconstruction in real-time. In: Intelligent Vehicles Symposium (IV) (2011)
8. Høydal, , Skytøen, E., Andersson, S., Moser, M.B., Moser, E.: Object-vector coding in the medial entorhinal cortex. Nature **568**, 1–8 (04 2019). https://doi.org/10.1038/s41586-019-1077-7
9. Kiggundu, A., Weber, C., Wermter, S.: A compressing auto-encoder as a developmental model of grid cells. Human Brain Project HBP Student Conference, Austria pp. 35–37 (2017)
10. Leordeanu, M., Hebert, M.: A spectral technique for correspondence problems using pairwise constraints. Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1 **2**, 1482–1489 Vol. 2 (2005)
11. Liu, Z., Amini, A., Zhu, S., Karaman, S., Han, S., Rus, D.: Efficient and robust lidar-based end-to-end navigation. 2021 IEEE International Conference on Robotics and Automation (ICRA) pp. 13247–13254 (2021)
12. Müller, S., Weber, C., Wermter, S.: RatSLAM on humanoids - a bio-inspired SLAM model adapted to a humanoid robot. In: ICANN. pp. 789–796 (2014)
13. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. ArXiv **abs/1804.02767** (2018)
14. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.R.: Orb: An efficient alternative to sift or surf. 2011 International Conference on Computer Vision pp. 2564–2571 (2011)
15. Yu, F., Shang, J., Hu, Y., Milford, M.: NeuroSLAM: a brain-inspired SLAM system for 3d environments. Biological Cybernetics **113**(5-6), 515–545 (December 2019). https://doi.org/10.1007/s00422-019-00806-9, https://eprints.qut.edu.au/198104/
16. Zhou, X., Weber, C., Wermter, S.: A self-organizing method for robot navigation based on learned place and head-direction cells. 2018 International Joint Conference on Neural Networks (IJCNN) pp. 1–8 (2018)