Honda Research Institute EU
**Honda Research Institute Europe GmbH**
https://www.honda-ri.de/

# Efficient Navigation Using Slow Feature Gradients

## Benjamin Metka, Mathias Franzius, Ute Bauer-Wersing

### 2017

**Preprint:**

This is an accepted article published in International Conference on Intelligent Robots and Systems (IROS). The final authenticated version is available online at: https://doi.org/10.1109/IROS.2017.8202307

# Efficient Navigation Using Slow Feature Gradients

Benjamin Metka[1] and Mathias Franzius[2] and Ute Bauer-Wersing[1]

***Abstract*— A model of hierarchical Slow Feature Analysis (SFA) enables a mobile robot to learn a spatial representation of its environment directly from images captured during a random walk. After the unsupervised learning phase a subset of the resulting representations are orientation invariant and code for the position of the robot. Hence, they change monotonically over space even though the variation of the sensory signals received from the environment might change drastically e.g. during rotation on the spot. Furthermore, the property of spatial smoothness allows us to infer a navigation direction by taking the difference between the measurement at the current location and a measurement at a target location. In our work we investigated the use of slow feature representations, learned for a specific environment, for the purpose of navigation. We present a straightforward method for navigation using gradient descent on the difference between two points specified in slow feature space. Due to its slowness objective, the resulting slow feature representations implicitly encode information about static obstacles, allowing a mobile robot to efficiently circumnavigate them by simply following the steepest gradient in slow feature space.**

## I. INTRODUCTION

Navigation is a crucial ability for autonomous mobile robots executing tasks in a spatial environment. To perform navigation a robot needs an internal representation of the environment to estimate its own location and to plan a viable path to a target. Methods for visual simultaneous localization and mapping enable a robot to build a map of the environment and estimate its location at the same time using vision as the only sensory input. The resulting maps represent the environment in different ways, e.g. as a graph structure representing the topology or a discretized occupancy grid, which leads to different navigation strategies [1]. These strategies have different levels of complexity ranging from reactive motion execution to path planning in metrical maps [2].

Navigation is one of the most challenging tasks for mobile robots. Many animals, on the other hand, have excellent navigation capabilities. The paths they take may be suboptimal, but they are rapidly selected, flexible and result in an adaptive and robust navigation behavior. In a rat's brain special neurons have been identified which encode spatial attributes of the animal. Place cells encode its position within the environment and head-direction cells encode its orientation [3]. It has been demonstrated that slowly varying features extracted from the visual input of a virtual rat can model place cells and head-direction cells [4], [5]. In slowness learning [6], the resulting representation strongly depends on the movement statistics of the animal. Position encoding with invariance to head direction requires a relatively large amount of head rotation around the yaw axis of the animal compared to translational movement during mapping of the environment. Recently, an extension to the model, using an uncalibrated omnidirectional imaging system, was successfully applied to a mobile robot in an outdoor environment [7]. Additional rotational movement is simulated during the learning phase by circular shifting the periodic panoramic images for improved orientation invariance.

Another model of rat navigation is RatSLAM [8]. The pose is encoded by an activity packet in a 3d continuous attractor network with axis representing $(x, y, \theta)$. Self-motion cues and visual template matching inject energy into the network shifting the peak of activity. An extension of RatSlam is the organization of unique combinations of local views and pose codes in a graph like experience map which enables the model to maintain a consistent spatial representation over extended periods of time [9].

Navigation in topological maps is relatively straightforward and can be performed using the graph search algorithm A* [10]. Given an admissible distance heuristic it is guaranteed to find the optimal path but it is memory and computationally intensive for large environments with many obstacles. The potential field method is a different approach for navigation in metrical space that is based on gradient descent in a vector force field defined by an attractor at the target position and repulsive forces from obstacles [11], [12]. Although it is an elegant solution, a known limitation of the approach are local minima caused by certain types of obstacles or their spatial configuration [13].

Reinforcement learning has been applied to the low dimensional representation of the environment, extracted from visual input using Slow Feature Analysis (SFA), to learn policies that guide an agent to a goal location in a simplified version of the Morris water maze task [14] and with views from a mobile robot [15]. The presented results demonstrate the feasibility of the approach but it requires an additional learning step for new goal locations.

We propose a new method for navigation in slow feature space using gradient descent which builds upon the orientation invariant representations of the location learned with the SFA-model proposed in [7]. After the unsupervised learning of the environmental representation, navigation can be performed efficiently by following the SFA-gradient, approximated from distance measurements between the target and the current value. Moreover, we experimentally show that the learned slow feature representations implicitly en-

[1]Benjamin Metka and Ute Bauer-Wersing are with the Faculty of Computer Science and Engineering, Frankfurt University of Applied Sciences, 60318 Frankfurt, Germany bmetka@fb2.fra-uas.de

[2]Mathias Franzius is with the Honda Research Institute Europe GmbH, 63073 Offenbach, Germany

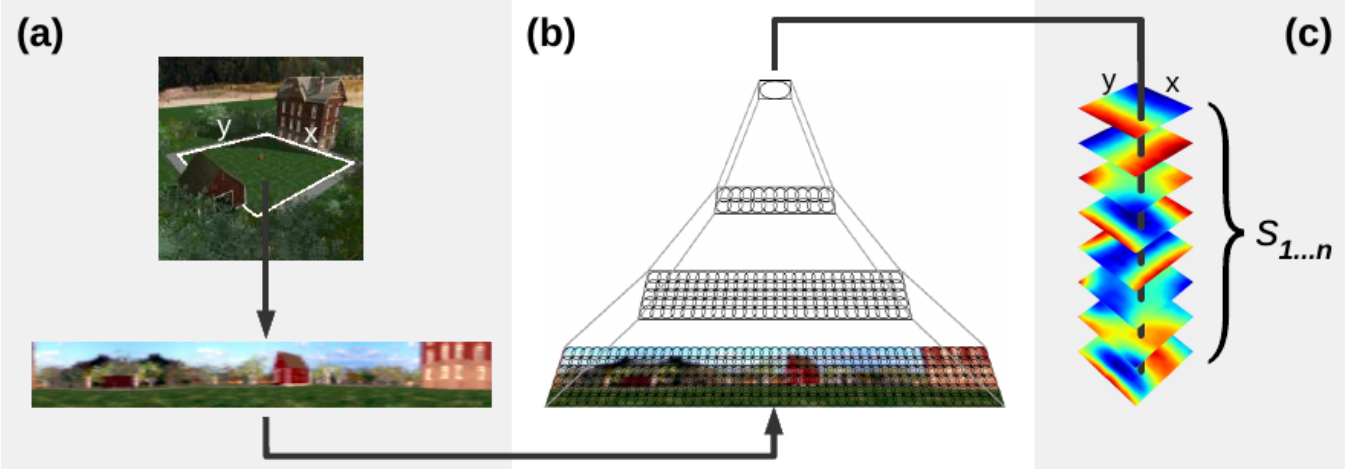



Fig. 1. **Learning the spatial representation**. (a) The omnidirectional view from the current position on the training trajectory is captured and transformed to a panoramic view. Since the resulting slow feature representations only depend on the statistics of the training data we simulate a full rotation for every image by circular shifting the panoramic view. This way we increase the perceived rotational movement which leads to an orientation invariant encoding of the position. (b) The views are processed by the network. Each layer consists of overlapping SFA-nodes arranged on a regular grid. Each node performs linear SFA for dimensionality reduction followed by a quadratic SFA for slow feature extraction. The output layer is a single SFA-node. (c) Ideally, the learned representations are orientation invariant and only code for the position. In our case the color-coded SFA-outputs of the last node $s_{1...n}$ over all positions, so-called *spatial firing maps*, show characteristic gradients along the coordinate axes.

code information about obstacles which are reflected in the gradients and lead to an efficient navigation which requires no explicit obstacle avoidance planning.

## II. Learning Spatial Representations with SFA

### A. Slow Feature Analysis

SFA as introduced in [6] transforms a multidimensional time series $\boldsymbol{x}(t)$, in our case images along a trajectory, to slowly varying output signals. The objective is to find instantaneous scalar input-output functions $g_j(\boldsymbol{x})$ such that the output signals

$$s_j(t) := g_j(\boldsymbol{x}(t))$$

minimize

$$\Delta(s_j) := \langle \dot{s}_j^2 \rangle_t$$

under the constraints

$$\begin{aligned} \langle s_j \rangle_t &= 0 \text{ (zero mean)}, \\ \langle s_j^2 \rangle_t &= 1 \text{ (unit variance)}, \\ \forall i < j : \langle s_i s_j \rangle_t &= 0 \text{ (decorrelation and order)} \end{aligned}$$

with $\langle \cdot \rangle_t$ and $\dot{s}$ indicating temporal averaging and the derivative of $s$, respectively. The $\Delta$-value is a measure of the temporal slowness of the signal $s_j(t)$. It is given by the mean square of the signal's temporal derivative, so small $\Delta$-values indicate slowly varying signals. The constraints avoid the trivial constant solution and ensure that different functions $g$ code for different aspects of the input. We use the MDP [16] implementation of SFA, which is based on solving a generalized eigenvalue problem.

### B. Learning Spatial Representations and Orientation Invariance

To learn representations that are suitable for the task of navigation we want to learn functions that encode the robot's position as slowly varying features and are invariant w.r.t. its orientation. The kind of spatial information encoded in the learned slow feature representations depends solely on the statistics of the training data. To achieve orientation invariance, the orientation of the robot has to change on a faster timescale than its position [5]. An omnidirectional camera is a feasible solution to simulate additional robot rotation by shifting a sliding window over the periodic panoramic images [7].

The high dimensional visual input is processed by a hierarchical network made of several converging layers. In our standard approach [7], layers are trained subsequently with all training images, and their shifted variants from the simulated rotation, in temporal order of their recording. A single node per layer is trained with stimuli from all node locations in its layer and replicated throughout the layer after training. The layers converge onto a single SFA-node whose slowest outputs $s_{1...n}$ form the slow feature representation of the environment we use for navigation. Once the training is finished, slow features are computed instantaneously from a single image. The learned spatial representations are analyzed qualitatively by plotting the color-coded slowest SFA-outputs computed for the images of all positions in the training area. In case of strong spatial coding a clear gradient along the coordinate axis is visible in the spatial firing maps of the first two units, since the theoretically optimal functions encode the $x, y$ coordinates as a half sine-/cosine-waves [5]. Maps of higher units show a mixture of the first two units or higher modes. The learning process and ideal spatial firing maps are illustrated in Fig. 1.

Due to the slowness objective of the SFA learning algorithm, input data that is temporarily close during training will be encoded with similar values while temporarily distant data will be separated in slow feature space. Since a wheeled mobile robot can not jump over an obstacle the mean temporal distance between points on opposite sides of the obstacle is larger than the temporal distance to nearby points on the same side. Thus, the learned representations implicitly encode information about obstacles present during the training, resulting in gradients of the slow feature values that "flow" around the obstacles.

## III. Navigation with Slow Feature Gradients

Applying the simulated rotation during the learning phase increases the relative amount of rotation compared to translation. Since the learned slowest feature representations are invariant with respect to the orientation of the robot and only code for its position, they change monotonically over space [7]. Given two points in 2d space we can take the difference between their slow feature representations to define a cost function and estimate a navigation direction by approximating the gradient of the cost surface. Navigation between a start and a target location can be achieved by performing gradient descent on the cost surface. The slow feature representations of the visual inputs at the target locations can be acquired during the training phase at points of interest e.g. the charging station of the robot.

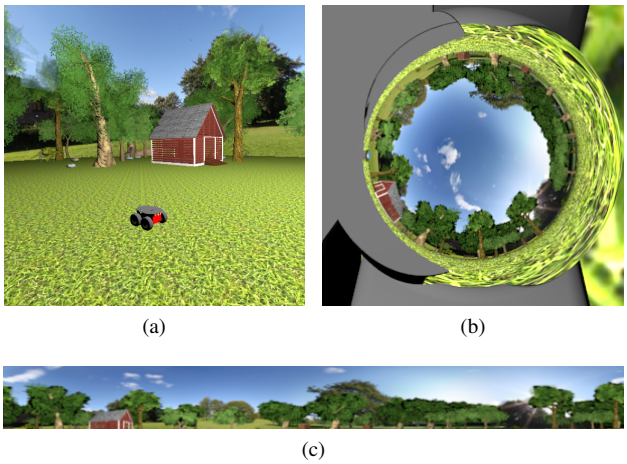
Assume that $n$ slow feature components have been chosen to be used for the representation of the environment. The current position in 2d space is given by $p := (x_p, y_p)$. The mapping function $f : \mathbb{R}^2 \mapsto \mathbb{R}^n$ transforms a position to the slow feature space by processing the corresponding image. Given a target position $t := (x_t, y_t)$, we define the function $C : \mathbb{R}^n \mapsto \mathbb{R}$ which computes the cost from $p$ to $t$, taking only $f(p)$ as input. $C$ can be any distance function such as the Euclidean distance:

$$C(f(p)) = \sqrt{\sum_{i=1}^{n} (f(p)_i - f(t)_i)^2}$$

Ideally, we require the analytic gradient $\frac{\partial C(f(p))}{\partial p}$ as navigation direction which is, however, infeasible to obtain. Therefore, we compute its local linear approximation. In addition to the cost of the current position $C(f(p))$ we acquire at least two additional cost values $C(f(p_1))$, $C(f(p_2))$ for points "close-by", which have to be non-collinear, and which are used to fit a plane to the surface of $C$.

### A. Implementation

In order to acquire the cost function measurements from at least three nearby positions in an efficient way, we place one omnidirectional camera on each side of the robot with a fixed offset to the robots coordinate system. Thus, we obtain two measurements for each time step. To estimate the first gradient at time $t_0$ we take the difference quotient from the cost values computed for the image from the left and the right camera and the corresponding translation vector. After the first step, along the estimated gradient we can measure two additional cost values at time $t_1$. Next we estimate the plane defined by the four points obtained at $t_0$ and $t_1$. Therefore, we take three points to define two vectors and compute the plane normal from their cross product. We repeat this step for all four possible combinations and take the mean of the normal vectors to approximate the gradient by the slope of the corresponding plane. Then we make a step along the gradient direction and replace the points from $t_0$ with those from $t_1$ and those from $t_1$ with the cost values measured at the new location. This process is repeated until a predefined precision is achieved or the maximum number of iterations is reached. The estimated gradient is multiplied with a scaling factor $\eta$. A momentum term $\gamma$ is used to incorporate information from past gradient information to improve convergence and overcome local minima.



(a) (b)

(c)

Fig. 2. **Simulated environment**. (a) Experiments were performed in simulated garden like environment. (b) Example image from the omnidirectional camera. One camera is attached to each side of the robot. (c) The field of view of the unwarped panoramic images is cropped to discard static image regions.

## IV. Experiments

The proposed navigation method was evaluated in experiments in a simulated garden like environment with a size of $18 \times 22$ units (for reference: the robot has a side length of $\approx 0.7$ units). We used the 3d-software Blender[1] and its python API for creating the environment and performing the experiments. The virtual robot is equipped with two omnidirectional cameras attached to its left and right side. Images from the omnidirectional cameras are rendered with a resolution of $300 \times 300$ pixel and unwarped to panoramic views with a resolution of $600 \times 60$ pixel. The vertical field of view of the panoramic images is reduced to discard the static parts of the image i.e. the other camera and the robot. The simulator environment as well as a rendered image from the omnidirectional camera and the unwarped panoramic view are illustrated in Fig. 2. During the training phase the robot starts to move along a line with a random orientation. In

[1] https://www.blender.org/

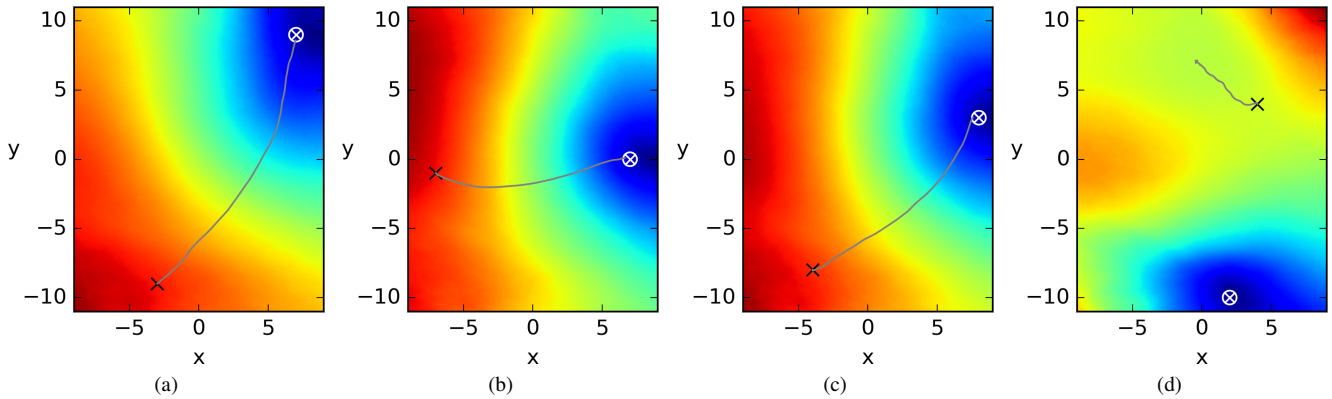



Fig. 4. **Resulting trajectories.** The start and target position are marked by a black cross and a white circle, respectively. (a)-(c) The robot successfully navigated to the target position performing gradient descent on the first two slow feature functions. (d) Using four slow feature functions leads to local minima on the cost surface since functions of later units will contain higher modes of previous ones.

case it reaches the border of the training area or the border of an obstacle a new orientation is chosen randomly and the robot follows the new direction. The velocity is kept constant with $0.2$ units per time step. We captured $5000$ images for the experiments to ensure an even sampling of the environment. Using more images increases the quality of the learned representations while a reasonable representation can be obtained with less images on a directed path with few crossing along each coordinate axis [5]. We set the gradient scaling factor $\eta = 1.0$ and the momentum $\gamma = 0.5$ for all experiments. The number of slow feature components $n$ is varied between one and six. The navigation task is considered successful if the robot ends up within a radius of $0.5$ units with respect to the specified target location. The maximum number of iterations is set to $300$. After the unsupervised learning phase we create the spatial firing maps $1...n$ by plotting the color-coded SFA-outputs $s_{1...n}$ for the images captured on a fixed grid. We create the plots of the cost-surface accordingly by plotting the color-coded cost from all grid positions to the target position in slow feature space. Please note that metric information is only used for the purpose of illustration and that the spatial representations are solely learned from the images.

### A. Navigation in an Open Field Scenario

To validate our approach we first tested the navigation method in an open field scenario. The random training trajectory can be seen in Fig. 3a. The spatial firing maps, shown in Fig. 3b, of the four slowest functions show strong spatial coding illustrated by the characteristic gradients along the coordinate axes of the first two functions. Function three is a mixture of the first two functions and function four is a higher mode of the second one. We performed $50$ trials with randomly chosen start and target points from within the training area so that the minimum distance between them amounts at least to $15$ units. The $50$ trials are repeated using an increasing number of slow features $s_{2...6}$. We evaluate the success rate and the efficiency of the trajectories. The efficiency is calculated as the ratio of the direct distance and

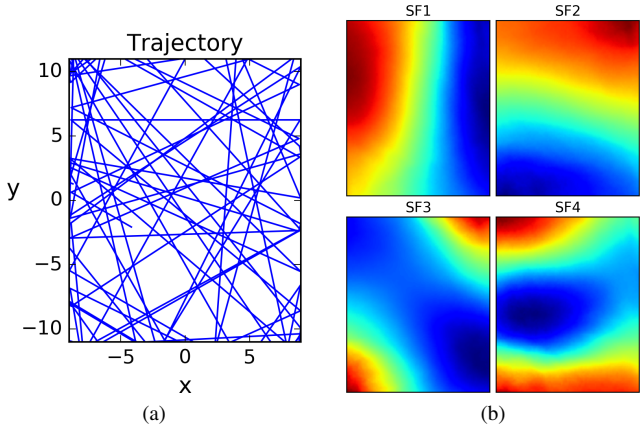



Fig. 3. (a) The training trajectory consists of $5000$ positions along line segments with random orientation. (b) Spatial firing maps of the first two SFA functions clearly encode the position along the coordinate axes with characteristic gradients. Function three is a mixture of the first two functions, whereas function four is a higher mode of the second one.

the traveled distance so that the highest efficiency value is one. The efficiency is only calculated for successful trials and then given as the average.

*1) Results:* The best navigation performance is achieved using the first two slow feature functions with a success rate of $0.96$ and an efficiency of $0.94$. Example trajectories are shown in Fig. 4. In the two cases where the navigation failed, the robot got obviously stuck in a local minimum close to the target location. The success rate significantly decreases from $0.96$ to $0.16$ with an increasing number of used slow feature functions, whereas the efficiency is slightly reduced from $0.94$ to $0.87$ for the successful trials (see Fig. 5). Since additional functions contain higher modes of preceding ones, the gradient descent is more likely to get stuck in local minima.

### B. Navigation with an Obstacle

For the following experiment we placed a v-shaped obstacle in the scene to validate the assumption that the slow feature representations implicitly encode information about

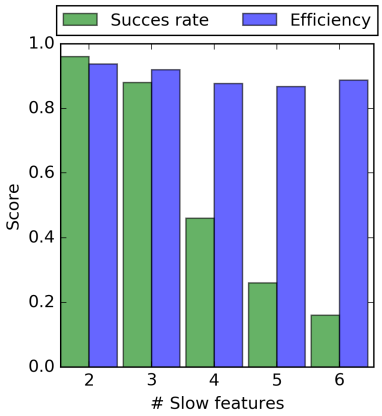



Fig. 5. **Navigation results for the open field scenario**. The navigation works best when using the first two slowest features with a success rate of 0.96 and an efficiency of 0.94. Using more features leads to a decreased performance since successive functions are higher modes of the previous ones leading to local minima.

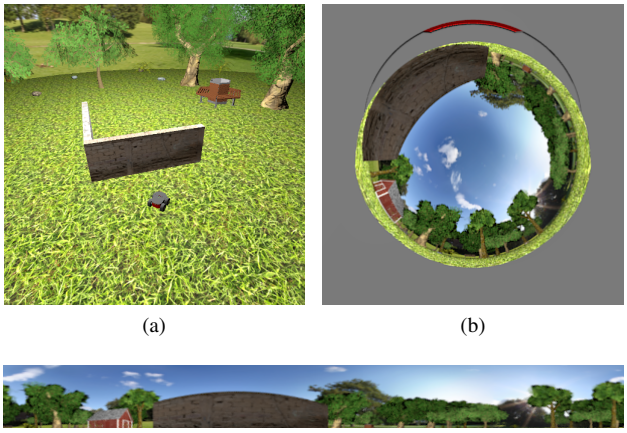



Fig. 6. **Simulated environment with an obstacle**. (a) A v-shaped obstacle placed in the simulated garden like environment. (b) Example image from the omnidirectional camera, one attached to each side of the robot. (c) The field of view of the unwarped panoramic images is cropped to discard static image regions.

obstacles in the scene and allow the robot to circumnavigate obstacles by simply following the steepest gradient. The simulator environment with an obstacle placed in the scene, as well as a rendered image from the omnidirectional camera and the unwarped panoramic view are illustrated in Fig. 6. The target location is kept fixed on the upper side of the obstacle, since it is the most interesting configuration, while the starting locations are randomly drawn from the lower half of the training area on the opposite side of the obstacle. Again we perform 50 trials but only consider SFA outputs $s_1$ and $s_2$, since we achieved the best performance using them in the previous experiment. The random training trajectory is shown in Fig. 7a. The spatial firing maps of the first four SFA-outputs are illustrated in Fig. 7b. The implicit encoding of the obstacle is clearly visible in the first two functions. To compute the efficiency of the resulting trajectories we discretized the training area into a grid with a cell size of 0.1 units and applied A* to obtain the optimal path.

*1) Results:* The robot reached the target location in 84% of the trials, successfully circumnavigating the obstacle following a nearly optimal trajectory with a mean efficiency of 76%. Examples of the resulting trajectories of the experiment are illustrated in Fig. 8. In case of a failure, the robot got stuck in a local minima. Since most of the variance is concentrated in regions near the obstacle the gradients for large parts of the training area are relatively flat. Using a more sophisticated method for gradient descent, these failure cases could be resolved.

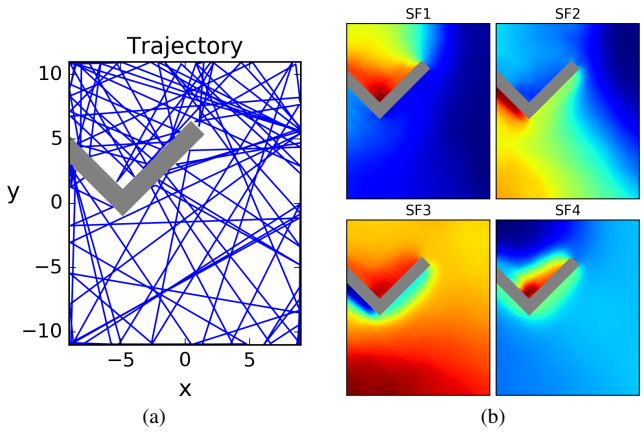



Fig. 7. (a) The training trajectory consists of 5000 positions along line segments with a random orientation. (b) The spatial firing map of the first SFA function contains the steepest gradient around the upper right corner of the obstacle where most of the variance is concentrated. The gradients in the spatial firing map of the second function are steepest around the lower corner of the obstacle. The spatial firing maps of the third and fourth function are difficult to interpret.

## V. Conclusion

We presented a straightforward and efficient method for navigating directly in slow feature space using gradient descent. The slow feature representations are learned for a specific environment in an offline learning phase where the robot randomly samples the environment. After the unsupervised learning step a navigation direction can be obtained very efficiently from three close-by measurements of the cost defined as the distance in slow feature space. Information about obstacles is implicitly encoded in the resulting slow feature representations which is reflected in the resulting gradients. Hence, circumnavigating obstacles requires no explicit planning of the trajectory but is accomplished by simply following the steepest gradient. In the simulator experiments the robot reached the target in 96% of the trials in an open field scenario and in 84% of the runs when the target location was behind an obstacle. In the failure cases the robot got stuck in regions with flat gradients. A more advanced gradient descent algorithm could cope with these cases and make the navigation more robust which would also be crucial for applying the method in real world scenarios.

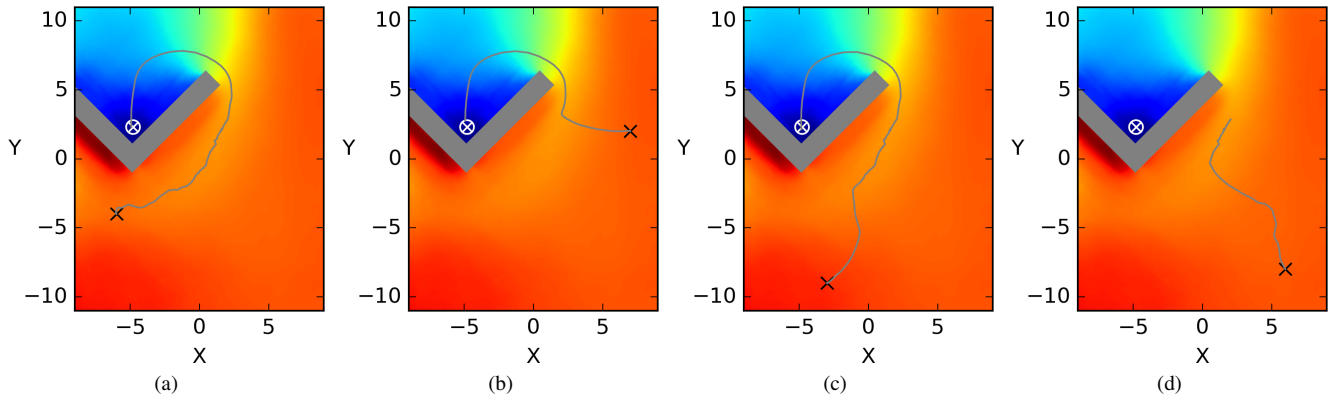



Fig. 8. **Resulting trajectories.** The start and target position are marked by a black cross and a white circle, respectively. (a)-(c) Using the first two slow feature functions enables the robot to successfully navigate around the obstacle to the target location. (d) Since most of the variance of the slow feature representations is concentrated near the obstacle the gradients in large parts of the training area are rather flat so that the robot got stuck.

## REFERENCES


[1] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55–81, 2015.

[2] J. Meyer and D. Filliat, "Map-based navigation in mobile robots. II. A review of map-learning and path-planning strategies," *Cognitive Systems Research*, vol. 4, no. 4, pp. 283–317, 2003.

[3] K. J. Jeffery and J. M. OKeefe, "Learned interaction of visual and idiothetic cues in the control of place field orientation," *Experimental Brain Research*, vol. 127, no. 2, pp. 151–161, 1999.

[4] R. Wyss, P. König, and P. F. M. J. Verschure, "A model of the ventral visual system based on temporal stability and local memory," *PLoS Biol*, vol. 4, p. e120, 04 2006.

[5] M. Franzius, H. Sprekeler, and L. Wiskott, "Slowness and sparseness lead to place, head-direction, and spatial-view cells," *PLoS Computational Biology*, vol. 3, no. 8, pp. 1–18, 08 2007.

[6] L. Wiskott and T. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural Computation*, vol. 14, no. 4, pp. 715–770, 2002.

[7] B. Metka, M. Franzius, and U. Bauer-Wersing, "Outdoor self-localization of a mobile robot using slow feature analysis," in *Neural Information Processing - 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part I*, 2013, pp. 249–256.

[8] M. Milford, G. Wyeth, and D. Prasser, "RatSLAM: a hippocampal model for simultaneous localization and mapping," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA 2004, April 26 - May 1, 2004, New Orleans, LA, USA*, 2004, pp. 403–408.

[9] M. Milford and G. Wyeth, "Persistent navigation and mapping using a biologically inspired SLAM system," *I. J. Robotics Res.*, vol. 29, no. 9, pp. 1131–1153, 2010.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[11] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, Mar 1985, pp. 500–505.

[12] J. Barraquand and J. Latombe, "Robot motion planning: A distributed representation approach," *I. J. Robotics Res.*, vol. 10, no. 6, pp. 628–649, 1991.

[13] R. B. Tilove, "Local obstacle avoidance for mobile robots based on the method of artificial potentials," in *Proceedings., IEEE International Conference on Robotics and Automation*, May 1990, pp. 566–571 vol.1.

[14] R. Legenstein, N. Wilbert, and L. Wiskott, "Reinforcement learning on slow features of high-dimensional input streams," *PLoS Computational Biology*, vol. 6, no. 8, pp. 1–13, 2010.

[15] W. Böhmer, S. Grünewälder, Y. Shen, M. Musial, and K. Obermayer, "Construction of approximation spaces for reinforcement learning," *Journal of Machine Learning Research*, vol. 14, no. 1, pp. 2067–2118, 2013.

[16] T. Zito, N. Wilbert, L. Wiskott, and P. Berkes, "Modular toolkit for data processing (mdp): a python data processing framework," *Front. Neuroinform.*, vol. 2, no. 8, 2009.