A neurologically inspired sequence processing model for mobile robot place recognition

Peer Neubert, Stefan Schubert, and Peter Protzel

***Abstract*—Visual place recognition is the problem of camera based localization of a robot given a database of images of known places, potentially under severe appearance changes (e.g. different weather or illumination). Typically, the current query images and the database images are sequences of consecutive images. In previous work, we proposed to adapt Hierarchical Temporal Memory (HTM), a biologically plausible model of sequence processing in the human neocortex to address this task. The previous work described the algorithmic steps and showed synthetic experimental results from simulation. This paper extends the approach to application on real world data based on a novel encoder for state-of-the-art image processing front-ends. The neurologically inspired approach is compared to several state-of-the-art algorithms on a variety of datasets and shows preferable performance. Beyond the place recognition performance, the neurological roots of the algorithm result in appealing properties like potentially very energy efficient implementation due to the usage of sparse distributed representations and natural extendability like the integration of motion estimates similar to entorhinal grid cells. Finally, we underline its practical applicability by online, soft real-time application on a mobile robot.**

***Index Terms*—Biologically-Inspired Robots, Neurorobotics, Localization, Visual-Based Navigation**

I. INTRODUCTION

VISUAL place recognition is an important subproblem in mobile robot navigation. It is the problem of matching the robot's current camera images to a set of images of known places. In particular, it is an important mean for loop closure detection in simultaneous localization and mapping (SLAM). The simplest visual place recognition approach is an individual, pairwise comparison of query images to images in the database. This is prone to errors, in particular in case of place aliasing (different places in the world share the same appearance) or changing environmental conditions (e.g. matching images between different lighting conditions). To address this problem, metric motion information can be used to iteratively estimate the current pose to support visual place recognition - ultimately, this means first creating a map (solving metric SLAM) and then using this map to perform place recognition. Since the incremental pose estimate can be in gross error [1], this might not be an option.

Fig. 1 shows the simple pairwise comparison and metric SLAM as the opposite directions of a spectrum of approaches.

Manuscript received: February 24, 2019; Revised: May 10, 2019; Accepted: June 10, 2019.

This paper was recommended for publication by Editor Cyrill Stachniss upon evaluation of the Associate Editor and Reviewers' comments.

The authors are with Faculty of Electrical Engineering and Information Technology, Technische Universität Chemnitz, Germany. firstname.lastname@etit.tu-chemnitz.de

Digital Object Identifier (DOI): see top of this page.

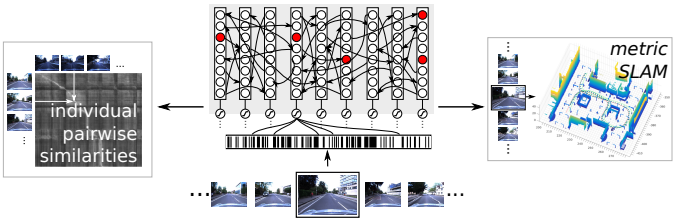

Fig. 1. Pairwise image matching and metric SLAM are at the opposite ends of a spectrum of approaches. We present a biologically inspired place recognition system in between. It is based on recurrent minicolumn networks that learn a topological map of sequential information in the input - including multiple paths with intersections, sequence repetitions and sequence variations.

Our work targets at approaches in between, that evaluate sequences of images to exploit some kind of systematics in the data to foster visual localization, but do not yet create a metric map. A well known example is SeqSLAM [2] which evaluates short sequences of images to decide about place matchings, but has the limitation of a non-repetitive trajectory with constant velocities between traversals.

To address the more complex trajectories of real world robots, e.g. with multiple revisits of the same place, stops, and path intersections, this work presents and evaluates a neurologically inspired approach to sequence processing for visual place recognition. Processing of sequence information is a fundamental capability of brains. Although detailed understanding of mechanisms in brains of even moderate size is still lacking, there are theories of particular aspects and working principles in brains. In this work, we take inspiration from Hierarchical Temporal Memory (HTM) [3], a theory on working principles of the human neocortex, in particular of predictive connections between cells in minicolumns. In previous work [4], we discussed analogies between HTM and the place recognition problem, proposed a simplified version of HTM's higher order sequence memory for place recognition and showed proof-of-concept results in simulation.

In this work, we extend this to application on real world data, in particular, in combination with CNN-based image descriptors. We recap the algorithmic approach from [4] in Sec. III. The first contribution of this paper is a concise presentation of the procedural algorithmic description in [4] in a set of equations in Sec. III-A. Sec. III-B presents a modified minicolumn reactivation strategy. Sec. III-C sets the stage for real world application by introducing a new visual front-end that encodes existing image descriptors like AlexNet-conv3 or NetVLAD for combination with the presented place recognition back-end. The experimental evaluation is three-fold: Sec. IV evaluates robustness to noise and parameters of the back-end in a controlled simulation environment, Sec. V

compares the combined back-end/front-end approach on a variety of long-term real-world datasets to several existing algorithms and Sec. VI demonstrates the online applicability with two shorter real-time experiments on a mobile robot. A Matlab implementation of the presented system is available.[1]

II. RELATED WORK

Visual place recognition is a well studied problem. Lowry et al. [5] provide a recent survey. Typically, a first step is to extract image descriptors, like local landmarks [6], [7] or global descriptors [8], [9]. For place recognition in changing environments, descriptors from early convolutional layers (e.g. conv3) from off-the-shelve CNNs like AlexNet [10] showed impressive results [11]. More recently, CNN descriptors were also particularly designed and trained for place recognition, e.g. NetVLAD [12]. In our experiments, we will use both CNN descriptor types, AlexNet and NetVLAD.

Based on such image processing front-ends, a variety of approaches exists to compare and match images. Beyond simple pairwise comparison and using statistics of feature appearances (e.g. FAB-MAP [13]), the benefit of exploiting sequence information is well accepted in the literature: For SeqSLAM, Milford et al. [2] proposed to search for linear segments of high similarity in the pairwise similarity matrix after a local contrast normalization. This approach significantly improved the state of the art at this time. To deal with limitations regarding traversal velocities, several extensions have been proposed. E.g., allowing non-zero acceleration [14] or searching for optimal paths in the similarity matrix using a graph-theoretical max-flow formulation [15]. Lynen et al. [16] propose a "placeless" place-recognition system based on continuous matching between image streams. Hansen and Browning [17] model sequence based place recognition as Hidden Markov Model and use the Viterbi algorithm to find a solution by dynamic programming. Arroyo et al. [18] use concatenated binary features to represent sequences. Vysotska et al. present a series of approaches to efficient place recognition using a graph theoretical approach [19] in combination with lazy data association [20] and hashing-based relocalization [21]. The proposed approach incrementally creates internal representations for places with unseen appearances, this is related to experience-based navigation [22]. RatSLAM [23] is a biologically inspired approach to SLAM that is inspired by entorhinal grid cells in the rat's brain. It uses a continuous attractor network (CAN) whose dynamics apply a temporal filter on the sensory data. In section V we will compare against [2], [17], [18], [19], [20], [21]. Another related approach to infer a camera's pose is to perform pose regression with neural networks. Kendall et al. [24] showed that a feed-forward convolutional neural network (GoogLeNet) trained with a simple loss function can be sufficient to regress position and orientation. Walch et al. [25] perform pose regression by feeding a CNN output vector to LSTMs. Clark et al. [26] use this combination of CNN and LSTM to exploit sequential information in video sequences.

[1]https://www.tu-chemnitz.de/etit/proaut/en/research/seqloc.html

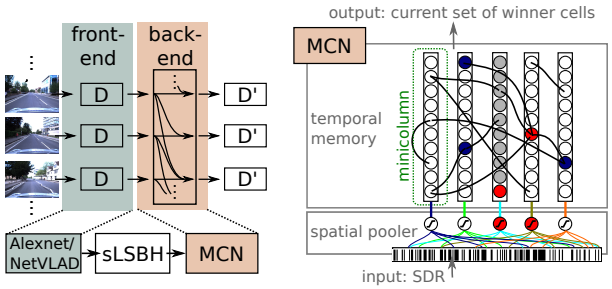

Fig. 2. *(left)* Overall approach. *(right)* Illustration of a minicolumn network (MCN). Red cells are current winner cells, grey cells are active (but not winner), blue cells are winner cells from the previous timestep.

This work builds upon our previous paper [4] which proposed the usage of Hierarchical Temporal Memory (HTM) for place recognition but did not work with any real world data. There, we mapped parts of HTM to aspects of the sequence-based place recognition problem and described an algorithmic approach on how to use a variant of the higher-order sequence memory algorithm from HTM for place recognition. HTM builds upon the assumption of a single learning algorithm that is deployed all over the neocortex [27]. It is a continuously evolving theory with the goal to find algorithmic formulations of working principles of the neocortex as well as extending the range of practical demonstrations and applications. Currently, these applications include anomaly detection, natural language processing, and object detection [28]. An open source implementation is available [29]. For more information on the biological background and HTM please refer to [30].

We want to emphasize the point that we do not claim that place recognition in human brains actually uses exactly the presented algorithm. There is plenty of evidence [31] of structures like entorhinal grid cells, place cells, head direction cells, speed cells and so on, that are involved in mammal navigation and are not regarded in this work. However, we believe that the presented algorithm can be naturally extended with concepts like grid cells as has been shown for HTM-based object recognition recently [32].

III. ALGORITHMIC APPROACH

The left part of Fig. 2 illustrates the overall approach: Each image is processed by a visual front-end that extracts a descriptor D. The back-end creates a new descriptor D' for each image based on the current descriptor and those of previous images. Descriptors D' of query and database image sequences can then be compared pairwise and still consider sequence information for each image.

To create descriptors D', we use a minicolumn network (MCN) that implements a simplified version of HTM's higher order sequence memory as described in [4]. This memory model is tightly coupled to insights about the neuroanatomic structure of the human neocortex: Sequential context is stored in cells that are arranged in cortical minicolumns and predict other cells through lateral connections, and the activation pattern of all cells forms a sparse distributed representation.

For this paper to be self-contained and to provide a different view on this algorithm, the following subsection reformulates our approach from [4] in a compact set of equations. For a

procedural algorithmic description and discussion of the theoretical background, please refer to [4]. To allow combination with real world image data, section III-B introduces a modified minicolumn activation strategy and section III-C describes a novel approach to create the input SDRs from images.

A. Simplified higher order memory in a minicolumn network

A minicolumn network (MCN) is a set of binary-state cells that are arranged in minicolumns. At each timestep, input is a binary sparse distributed representation (SDR), the output is the current state of the network (in particular, the set of winner cells). Fig. 2 illustrates the two parts of a MCN: (1) The spatial pooler which activates minicolumns that show a certain pattern and (2) the temporal memory that predicts and activates individual cells inside active minicolumns to distinguish different temporal context of this pattern. Learning is implemented by adding new minicolumns and new predictive connections between cells.

1) Spatial Pooler: The spatial pooler is a feed-forward network that processes the current input SDR. It reactivates minicolumns that correspond to known patterns and creates new minicolumns if there are not enough active minicolumns. For visual place recognition, it acts as feature detector for the current input image (respectively its SDR).

Minicolumn activation: Each minicolumn C_j has a binary state variable "active" A_j and a sparse set of feed-forward connections F_j from dimensions of the input SDR I. It becomes active at timestep t if sufficiently many of the connected input dimensions are active:

$$A_j^t = 1 \Leftrightarrow \sum_{m \in F_j} I_m^t \geq \theta \tag{1}$$

If less than k_{min} minicolumns are active, missing minicolumns are newly created; each samples an individual set of feed-forward connections F from the set of currently active dimensions in I^t. The MCN starts with an empty set of minicolumns and iteratively creates new minicolumns that respond to new patterns in the input SDR. Whenever this pattern reappears, the same minicolumns shall be activated. Using such an ensemble of k_{min} active minicolumns (each looking at a different part of the input by using an individual F_j) increases robustness to changes in I.

2) Temporal Memory: The second part of a MCN is the temporal memory. It is a recurrent binary-state network that accumulates information over consecutive input SDRs to encode the temporal context of the current SDR. This context information is encoded using different cells in active minicolumns. Each cell $c_{i,j}$ (the i-th cell in the j-th minicolumn) has three binary state variables: "active" $a_{i,j}$, "predicted" $p_{i,j}$, and "winner" $w_{i,j}$

Cell activation: Precondition for a cell $c_{i,j}$ to become active at time t is that its minicolumn C_j is active and that either particularly this cell was predicted ($p_{i,j}^t = 1$) or that none of the cells in this minicolumn is predicted (in HTM theory this is called "bursting"):

$$a_{i,j}^t = 1 \Leftrightarrow A_j^t \wedge (p_{i,j}^t \vee (\forall m : \neg p_{m,j}^t)) \tag{2}$$

Cell prediction: The "predicted" variable $p_{i,j}^t$ is set if there was an active cell $c_{m,n}$ at time $(t-1)$ with a lateral predictive connection to this cell $c_{i,j}$ in the set of directed lateral predictions $\mathbb{P}$ with $\mathbb{P} \subset \{(c_{m,n}, c_{i,j}) : i,j,m,n \in \mathbb{N}\}$:

$$p_{i,j}^t = 1 \Leftrightarrow \exists m,n : a_{m,n}^{t-1} \wedge (c_{m,n}, c_{i,j}) \in \mathbb{P} \tag{3}$$

Each iteration, the set of predictive connections is updated based on the previous and current winner cells. If there is already a connection from any cell in the previous minicolumn to the current winner cell, no new connection is created:

$$\mathbb{P} = \mathbb{P} \cup \{(c_{m,n}, c_{i,j}) : w_{m,n}^{t-1} \wedge w_{i,j}^t \wedge (\nexists l : (c_{l,n}, c_{i,j}) \in \mathbb{P})\} \tag{4}$$

Winner cells: Winner cells are active cells that were predicted. Additionally, in case of bursting (a minicolumn is active, but none of its cells is predicted, cf. eq. 2), a single random winner cell is selected from the bursting minicolumn; in the following equation, this event is indicated by random variable $b_{i,j}$ with $b_{i,j} = 1 \Leftrightarrow$ (C_j bursts and $c_{i,j}$ is randomly selected as winner):

$$w_{i,j}^t = 1 \Leftrightarrow (a_{i,j}^t \wedge p_{i,j}^t) \vee b_{i,j} \tag{5}$$

For more details and a procedural algorithmic description please refer to our previous work [4] and its open source implementation.

B. Modified reactivation strategy: $k_{max}NN$

A MCN can maintain different contexts for different cells in active minicolumns to prevent false-positive place associations. To achieve a low false-negative rate (i.e. to not miss place matchings), the appropriate reactivation of minicolumns is crucial. In the proof-of-concept demonstration in [4], we used a simple threshold θ on the overlap similarity between minicolumn and input pattern (cf. eq. 1). The choice of this threshold has large influence on the results. To reduce this influence, in this work, we additionally restrict the reactivation with a k_{max}-nearest-neighbor ($k_{max}NN$) condition: A minicolumn is reactivated iff. its overlap similarity to the input is $> \theta$ and not smaller than the overlap similarity of the k_{max}-th most similar minicolumn. This allows to use a less restrictive threshold θ without exhaustively reactivating minicolumns. A comparison of the two reactivation strategies (θ*-only* and θ+$k_{max}NN$) is shown in Fig. 3 (bottom-left).

C. SDRs and sparsified binary image encodings (sLSBH)

Beside temporal prediction in sequences, the second fundamental concept in HTM is the usage of Sparse Distributed Representations (SDRs). SDRs are high dimensional binary vectors (e.g. 8,192 dimensional) with very few 1-bits (e.g. 2.5%). There is evidence that SDRs are a widely used representation in brains due to their representation capacity, robustness to noise and power efficiency [33]. They are a special case of hypervector encodings, which have been used, e.g., to model memory [34], do approximate inference [35] and to learn simple robot behavior by imitation learning [36].

In MCN, an ensemble of at least k_{min} minicolumns is active in parallel. Since this is typically only a small fraction of all

minicolumns, this forms a sparse representation. To additionally distribute information across the active minicolumns, each minicolumn samples an individual set of connections F_j from the input pattern. MCN requires binary input data. Although the HTM's spatial pooler can further sparsify the input data, it benefits from an already low number of ones in the input data [37].

In general, any sufficiently sparse binary descriptor can be used as input for MCN. Based on their results in pairwise image comparison for place recognition, we want to build upon CNN-based descriptors like AlexNet [10] (i.e. the output of the conv3 layer) and NetVLAD [12]. Our encoding is a sparsified binary adaptation of locality sensitive hashing (LSH) based on random projections. It consists of the following two steps:

1. Random projection For using different feature descriptors like the above CNNs in MCN, we want to be able to control their number of dimensions and the distribution of information across the descriptor dimensions. The statistical properties of the two chosen CNN descriptors are quite different. AlexNet-conv3 descriptor has $n = 64,896$ dimensions and concatenates feature map values of local image regions which are expected to be highly correlated. In contrast, NetVLAD has $n = 4,096$ dimensions, accumulates residuals to words from a dictionary and creates a whitened distribution with decorrelated dimensions. We use random projections to control both properties. Given a normalized n-dimensional feature descriptor $x \in \mathbb{R}^n$ (e.g., from one of the above CNNs), we first project this vector to a high dimensional space using a row-wise L2-normalized random projection matrix $P \in \mathbb{R}^{m \times n}$, each entry in this matrix is sampled iid. from a standard normal distribution: $P_{i,j} \sim \mathcal{N}(0,1)$. In our experiments, we chose $m = 2^{14} = 16,384$. This results in approximately uniform angular distribution of the m row vectors of P in the n-dimensional hyperspace and creates a distributed (holographic) representation. The result of the projection is $y = P \cdot x$.

2. Binarization and sparsification The input to MCN is expected to be a sparse binary vector [37]. To create a binary vector from the result of the random projection y, the authors of [11] propose to only keep the sign information for each dimension. The sign of dimension k encodes on which side of the hyperplane with normal vector according to the k-th row in P the vector x ends. Hamming distance based on this sign information approximates the cosine distance of the original vectors [38]. The authors of [11] achieved 95 % place recognition accuracy using $m = 8,192$ bit. However, the sign vector is a *dense* binary vector. Instead of using the sign, we propose to create a *sparsified* binary LSH vector z by concatenating the binary vectors with ones at the s percent smallest absolute angles to the m plane normals and the s percent largest absolute angles:

$$z = [z^+ z^-]^T \text{ with} \tag{6}$$

$$z_i^+ = \begin{cases} 1 & \text{if } y_i \text{ amongst } \frac{s}{100} \cdot m \text{ largest values in } y \\ 0 & \text{otherwise} \end{cases}$$

$$z_i^- = \begin{cases} 1 & \text{if } y_i \text{ amongst } \frac{s}{100} \cdot m \text{ smallest values in } y \\ 0 & \text{otherwise} \end{cases}$$

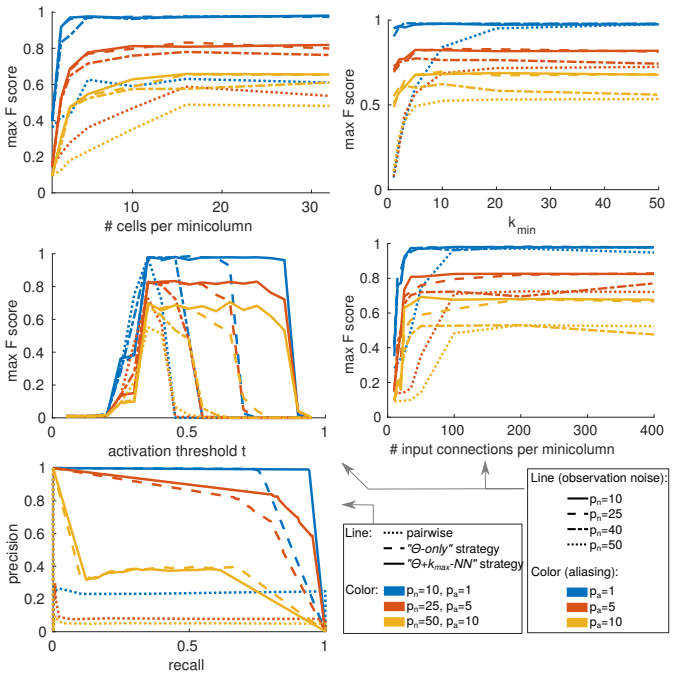

Fig. 3. Parameter evaluation using simulation. In the two top rows, the color encodes the amount of aliasing and the line style the observation noise.

The sparsified binary LSH (sLSBH) vector $z \in \{0,1\}^{2m}$ is the input to the MCN. The results section V will evaluate the accuracy of this encoding and compare the two CNN descriptors (AlexNet and NetVLAD) in combination with MCN and other place recognition approaches.

IV. Noise Tolerance and Parameter Evaluation

The first sequence of experiments evaluates noise tolerance and parameter settings under controlled conditions. We use a simple simulation environment following [4] that simulates a path in a grid-like world. Each grid cell is considered a place and the robot path is a sequence of neighbored places. The simulated sequence comprises 642 steps with multiple loops over 206 different places. For each place an 8,192 dimensional SDR is stored. At each visit of a place an observation measurement of its SDR is simulated. The measurements are affected by two types of noise, controlled by parameters p_n and p_a: (1) Observation noise: Each time a place is visited and observed, each of its SDR's 1-bits is randomly shifted to a wrong position with probability p_n. E.g., $p_n = 25\%$ indicates that on average 25% of the 1-bits for each observation of a place are shifted. (2) Place aliasing: Different places can share the same SDR (i.e., they can only be distinguished in the sequence context). Parameter p_a sets the probability of two randomly chosen places to share the same SDR. E.g., $p_a = 1\%$ implies that there are only 67 different SDRs to encode the 206 different places (accordingly: $p_a = 5\%$ implies 18 SDRs, $p_a = 10\%$ implies 10 SDRs for 206 places).

Each experimental run start with an empty minicolumn network. The SDR output from the simulator is iteratively fed into the system. For each timestep, the current set of MCN winner cells is stored. Place matchings are obtained

from the overlap of winner cell sets from different timesteps. The ground truth place matchings provided by the simulation environment are used to compute precision-recall curves from winner cell overlaps. To evaluate the influence of different amounts of noise, multiple traversals along the same path with the accordingly set noise parameters are simulated. The default MCN parameter setting for simulation experiments are $\theta = 0.4$, $k_{min} = 5$, $k_{max} = 10$, SDR length is 8,192, number of cells per minicolumn is 32 and there are 200 connections from the input to each minicolumn. To evaluate the influence of different parameter values, each traversal is evaluated multiple times while varying a single MCN parameter.

For each MCN configuration, the maximum F1-score from the resulting precision-recall curves results in one point on a curve in Fig. 3. The bottom left plot shows the benefit from the k_{max}-NN minicolumn activation strategy. The performance of place recognition using direct pairwise comparison of the SDR observations from the simulation illustrates the general difficulty of the simulated place recognition task. Even for the lowest considered amount of noise ($p_n = 10$, $p_a = 1$), the pairwise comparison shows severe problems.

The evaluation of the number of cells per minicolumn shows that for increasing amount of aliasing in the environment, more cells are required to encode the different contexts (the yellow curves converge at higher parameter values than e.g. the blue curves). For the number of cells, the number of connections of a minicolumn to the input SDR, and the number of enforced active minicolumns k_{min}, higher values are in general better. It can be seen, that the higher the amount of observation noise (p_n), the higher is the required number of connections to the input. The 2.5% sparsity (roughly 200 of 8,192 dimensions) taken from the biological model is a reasonable value. The ratio of k_{max} to k_{min} has only minor influence for a large range of values (not shown).

The minicolumn activation threshold t has to be chosen carefully. Too low values cause activation of too many minicolumns. This effect is mitigated by the k_{max}-NN sampling strategy but not eliminated. Too high values hamper recall in case of high observation noise. For the used simulation, the value $t = 0.35$ results in best performance for all evaluated amounts of noise. However, this is a statistical effect of the mutual independence of the input SDR dimensions. Presumably, for real world data, there is no single best choice, but the parameter has to be adapted to the used front-end and maybe also to the environment. Our simulation results indicate that in case of varying and unknown amounts of noise, a too high threshold value is presumably less harmful than a too low value.

V. Comparison on real world datasets

This section compares MCN to other available back-end approaches. Therefore we modified or reimplemented available sequence processing place recognition algorithms such that all work with the same visual front-ends (NetVLAD and AlexNet). The baseline is a **Pairwise** comparison of individual descriptors using cosine distance for AlexNet and Euclidean distance for NetVLAD. For **SeqSLAM** [2] we used OpenSeqSLAM [39] and replaced the internally computed image similarity matrix with the pairwise cosine similarity matrix of the CNN descriptors and use standard parameter $d_s = 10$. We reimplemented **ABLE** [18] based on information from the paper and available code and replaced the concatenated binary descriptors with concatenated CNN descriptors. Sequence length is equal to SeqSLAM. For **ISM** [19], **OPR** [20] and **VPR** [21] we used the implementations by the authors with their default parameters but used precomputed Pairwise distances as cost matrix. Since runtime is not an issue for the compared approaches, we replaced the hash-based relocalization in VPR to also use the provided cost matrix. We reimplemented **HMM** [17] based on the information from the paper and Matlab's Viterbi implementation. Again, we used the same sequence length as SeqSLAM. For NetVLAD [12], we use the authors' version using VGG-16 and whitening trained on the Pitts30k dataset. For AlexNet [10], we use Matlab's ImageNet model.

The comparison is based on five datasets with different characteristics regarding environment, appearance changes, single or multiple visits of places, possible stops, or viewpoint changes. **StLucia** (Various Times of the Day) [40]: Collected with a forward facing webcam mounted on a car driving in a suburb between morning and afternoon over several days. Each sequence contains several loop closures. We sampled images at 1Hz. **Oxford** RobotCar [41]: Recorded with a car equipped with several cameras and lidars over a period of over a year. The dataset is demanding due to seasonal changes, weather, long-term changes like roadworks and building construction, a few loop closures within each sequence and stops in front of traffic lights or intersections. We sampled images at 10Hz of the front facing part of the trinocular stereo camera. **CMU** Visual Localization [42]: Five car rides along a 8km route with possible stops, weather, and seasonal changes. There are no loop closures witin a sequence. We use the left camera. **Nordland** [39]: Time and viewpoint synchronized rides along a single train track once in each season. We use the same image set as [7]. **Gardens Point** Walking [43]: Hand held camera on a single route on campus, two times at day and once at night with controlled viewpoint deviations. The dataset is special due to the outdoor/indoor location, many pedestrians, and severe lighting changes. Nordland and Gardens Point are time synchronized, for all other datasets, we used GPS for ground truth.

The experimental results of NetVLAD+sLSBH+MCN and AlexNet+sLSBH+MCN on the above datasets and the comparison to the above algorithms (using the same visual front-end) in table I are a main contribution of the paper. For all datasets, the *same* parameter setup for MCN is used ($k_{min} = 50$, $k_{max} = 100$, number of cells per minicolumn is 32). For the different visual front-ends, we only changed minicolumn activation threshold θ and the number of connections per minicolumn from 0.4 and 800 for NetVLAD to 0.75 and 200 for AlexNet. MCN is trained on the database sequence; during query, no new minicolumns are created. For each dataset-algorithm pairing, we compute precision-recall curves and report average precision in table I. Average precision is the area under the precision recall curve obtained by trapezoidal numerical integration. The bold numbers indicate the best

TABLE I

EXPERIMENTAL RESULTS FOR MCN WITH NETVLAD OR ALEXNET FRONT-END. METRIC IS AVERAGE PRECISION. BEST VALUES PER ROW ARE BOLD. THE COLORED ARROWS INDICATE LARGE ($\geq$25% BETTER/WORSE) OR MEDIUM ($\geq$10%) DEVIATION COMPARED TO "PAIRWISE". THE SLSBH COLUMN SHOWS THE RESULTS OF PAIRWISE COMPARISON OF SLSBH ENCODINGS OF FRONT-END DESCRIPTORS. MCN USES SLSBH AS INPUT.

	Dataset	DB	Query	Pairwise	ABLE	ISM	OPR	VPR	HMM	SeqSLAM	sLSBH	MCN (ours)
NetVLAD front-end	StLucia	10th Sep 08.45	19th Aug 08.45	0.51	0.48 →	0.54 →	0.53 →	0.47 →	0.54 →	0.13 ↓	0.49 →	**0.60** ↗
		10th Sep 10.00	21st Aug 10.00	0.58	0.45 ↘	0.54 →	0.54 →	0.49 ↘	0.55 →	0.12 ↓	0.57 →	**0.65** ↗
		10th Sep 12.10	21st Aug 12.10	0.60	0.43 ↓	0.53 ↘	0.53 ↘	0.47 ↘	0.51 ↘	0.12 ↓	0.59 →	**0.65** →
		10th Sep 14.10	19th Aug 14.10	0.45	0.49 →	0.53 ↗	0.53 ↗	0.46 →	**0.54** ↗	0.14 ↓	0.44 →	**0.54** ↗
		10th Sep 15.45	18th Aug 15.45	0.42	0.49 ↗	**0.51** ↗	0.50 ↗	0.42 →	0.48 ↗	0.13 ↓	0.40 →	0.49 ↗
	Oxford	9th Dec 14	16th Dec 14	0.87	0.76 ↘	0.53 ↓	0.03 ↓	0.51 ↓	0.18 ↓	0.05 ↓	0.86 →	**0.88** →
		9th Dec 14	3rd Feb 15	**0.93**	0.82 ↘	0.51 ↓	0.03 ↓	0.52 ↓	0.13 ↓	0.03 ↓	0.92 →	0.91 →
		9th Dec 14	19th May 15	0.83	0.66 ↘	0.55 ↓	0.01 ↓	0.54 ↓	0.18 ↓	0.05 ↓	0.82 →	**0.95** ↗
		19th May 15	3rd Feb 15	**0.85**	0.82 →	0.48 ↓	0.05 ↓	0.45 ↓	0.11 ↓	0.02 ↓	0.84 →	**0.85** →
	CMU	21th Apr 11	1st Sep 10	0.59	0.49 ↘	0.38 ↓	0.26 ↓	0.50 ↘	0.29 ↓	0.04 ↓	0.82 ↑	**0.86** ↑
			15th Sep 10	0.66	0.53 ↘	0.35 ↓	0.53 ↘	0.49 ↓	0.32 ↓	0.08 ↓	**0.81** ↗	**0.81** ↗
			21th Dec 10	0.41	0.36 ↘	0.34 ↘	0.51 ↗	0.46 ↗	0.29 ↓	0.06 ↓	0.56 ↑	**0.60** ↑
			2nd Feb 11	0.45	0.26 ↓	0.39 ↘	0.42 →	0.48 →	0.31 ↓	0.14 ↓	0.68 ↑	**0.85** ↑
	Nordland	fall	spring	0.39	**0.98** ↑	0.91 ↑	0.90 ↑	0.68 ↑	0.89 ↑	0.93 ↑	0.39 →	0.52 ↑
		fall	winter	0.06	0.73 ↑	0.19 ↑	0.17 ↑	0.26 ↑	0.59 ↑	**0.80** ↑	0.05 →	0.21 ↑
		spring	winter	0.11	0.85 ↑	0.75 ↑	0.18 ↑	0.39 ↑	0.72 ↑	**0.88** ↑	0.11 →	0.23 ↑
		winter	spring	0.11	0.85 ↑	0.46 ↑	0.39 ↑	0.32 ↑	0.73 ↑	**0.88** ↑	0.11 →	0.29 ↑
		summer	spring	0.32	**0.97** ↑	0.88 ↑	0.73 ↑	0.64 ↑	0.89 ↑	0.94 ↑	0.32 →	0.44 ↑
		summer	fall	0.63	**1.00** ↑	0.97 ↑	0.97 ↑	0.89 ↑	0.94 ↑	0.97 ↑	0.63 →	0.53 ↘
	Gardens Point	day left	night right	0.41	**0.79** ↑	0.61 ↑	0.48 ↗	0.29 ↓	0.02 ↓	0.15 ↓	0.39 →	0.43 →
		day right	day left	0.98	**1.00** →	0.69 ↓	0.69 ↓	0.69 ↓	0.33 ↓	0.68 ↓	0.98 →	0.99 →
		day right	night right	0.52	**0.80** ↑	0.64 ↗	0.64 ↗	0.47 →	0.20 ↓	0.30 ↓	0.49 →	0.54 →

	Dataset	DB	Query	Pairwise	ABLE	ISM	OPR	VPR	HMM	SeqSLAM	sLSBH	MCN (ours)
AlexNet front-end	StLucia	10th Sep 08.45	19th Aug 08.45	0.60	0.44 ↓	0.57 →	0.57 →	0.54 ↘	0.56 →	0.13 ↓	0.60 →	**0.61** →
		10th Sep 10.00	21st Aug 10.00	0.56	0.39 ↓	0.56 →	0.56 →	0.54 →	0.55 →	0.13 ↓	0.56 →	**0.62** ↗
		10th Sep 12.10	21st Aug 12.10	0.54	0.36 ↓	0.56 →	0.56 →	0.52 →	0.54 →	0.40 ↓	0.54 →	**0.63** ↗
		10th Sep 14.10	19th Aug 14.10	0.60	0.48 ↘	0.58 →	0.57 →	0.55 →	0.56 →	0.14 ↓	0.59 →	**0.62** →
		10th Sep 15.45	18th Aug 15.45	0.59	0.48 ↘	0.58 →	0.58 →	0.54 →	0.57 →	0.14 ↓	0.59 →	**0.64** →
	Oxford	9th Dec 14	16th Dec 14	0.48	0.32 ↓	**0.53** ↗	0.02 ↓	0.44 →	0.11 ↓	0.04 ↓	0.46 →	0.50 →
		9th Dec 14	3rd Feb 15	0.64	0.57 →	0.49 ↘	0.04 ↓	0.49 ↘	0.14 ↓	0.02 ↓	0.60 →	**0.75** ↗
		9th Dec 14	19th May 15	0.25	0.16 ↓	0.53 ↑	0.01 ↓	0.46 ↑	0.10 ↓	0.18 ↓	0.24 →	**0.86** ↑
		19th May 15	3rd Feb 15	0.36	0.41 ↗	0.48 ↑	0.05 ↓	0.41 ↗	0.10 ↓	0.02 ↓	0.30 ↘	**0.86** ↑
	CMU	21th Apr 11	1st Sep 10	0.39	0.30 ↘	0.39 →	0.51 ↑	0.46 ↗	0.27 ↓	0.04 ↓	0.51 ↑	**0.61** ↑
			15th Sep 10	0.57	0.45 ↘	0.35 ↓	0.17 ↓	0.45 ↘	0.30 ↓	0.07 ↓	**0.64** ↗	0.58 →
			21th Dec 10	0.29	0.28 →	0.35 ↗	0.26 →	0.37 ↑	0.30 →	0.06 ↓	0.34 ↗	**0.51** ↑
			2nd Feb 11	0.28	0.15 ↓	0.39 ↑	0.36 ↑	0.42 ↑	0.32 ↗	0.14 ↓	0.37 ↑	**0.65** ↑
	Nordland	fall	spring	0.81	**1.00** ↗	**1.00** ↗	0.99 ↗	0.91 ↗	0.96 ↗	0.98 ↗	0.81 →	0.94 ↗
		fall	winter	0.65	**0.99** ↑	0.98 ↑	0.98 ↑	0.87 ↑	0.94 ↑	0.98 ↑	0.64 →	0.82 ↑
		spring	winter	0.60	**0.99** ↑	0.98 ↑	0.98 ↑	0.90 ↑	0.94 ↑	0.98 ↑	0.59 →	0.82 ↑
		winter	spring	0.60	**0.99** ↑	0.96 ↑	0.96 ↑	0.89 ↑	0.95 ↑	0.98 ↑	0.59 →	0.81 ↑
		summer	spring	0.76	**1.00** ↑	0.99 ↑	0.98 ↑	0.89 ↗	0.94 ↗	0.98 ↑	0.75 →	0.92 ↗
		summer	fall	0.93	**1.00** →	**1.00** →	**1.00** →	0.99 →	0.95 →	0.98 →	0.93 →	0.98 →
	Gardens Point	day left	night right	0.11	0.21 ↑	0.55 ↑	**0.57** ↑	0.19 ↑	0.02 ↓	0.06 ↓	0.11 →	0.19 ↑
		day right	day left	0.60	**0.89** ↑	0.62 →	0.62 →	0.52 ↘	0.14 ↓	0.10 ↓	0.60 →	0.61 →
		day right	night right	0.53	**0.77** ↑	0.71 ↑	0.71 ↑	0.62 ↗	0.46 ↘	0.43 ↘	0.53 →	0.74 ↑

results per row. It can be seen that no algorithm is the single best one, but for most datasets and both front-ends the combination with MCN provides the best results. For single sequence datasets like Nordland, algorithms that exploit this apriori knowledge like SeqSLAM or ABLE perform superior. However, they perform worse if this assumption is violated (e.g. StLucia or Oxford). Interestingly, no algorithm is *always* better than the pairwise comparison, although all datasets provide sequence information. However, MCN is the algorithm with the best worst-case performance across all datasets. In most cases it is much better than Pairwise and in worst case it looses 16% compared to Pairwise (Nordland summer-fall with NetVLAD) which is significantly better than, e.g. ABLE with up to 46% loss on multiple datasets. MCN can represent arbitrary complex routes with repetitions and handle arbitrarily many matches within a sequence. Algorithms like SeqSLAM and VPR require additional algorithmic steps to detect multiple matchings (which are not provided in their available implementations). The comparison of the sLSBH and Pairwise columns shows the feasibility of the proposed sLSBH encoding. Surprisingly, on the CMU dataset pairwise matching based on sLSBH is even better than using the original descriptors.

VI. ONLINE EXPERIMENTS

In these experiments, the MCN runs online on a mobile robot and computes place associations in soft real-time (the onboard computer provided relocalizations as fast as possible) while the robot is moving around. The online character reduces the evaluation possibilities (i.e. we cannot compare against other algorithms). These experiments are mainly intended to demonstrate the feasible runtime and to underline the potential for practical application. All computations are done onboard on a laptop with i7-8550U CPU. The robot is equipped with a 250° aperture angle camera as well as 2D lidar, odometry and IMU; ROS is used for drivers and communication. Solely camera images are used for place recognition whereas other sensors are used for ground truth.

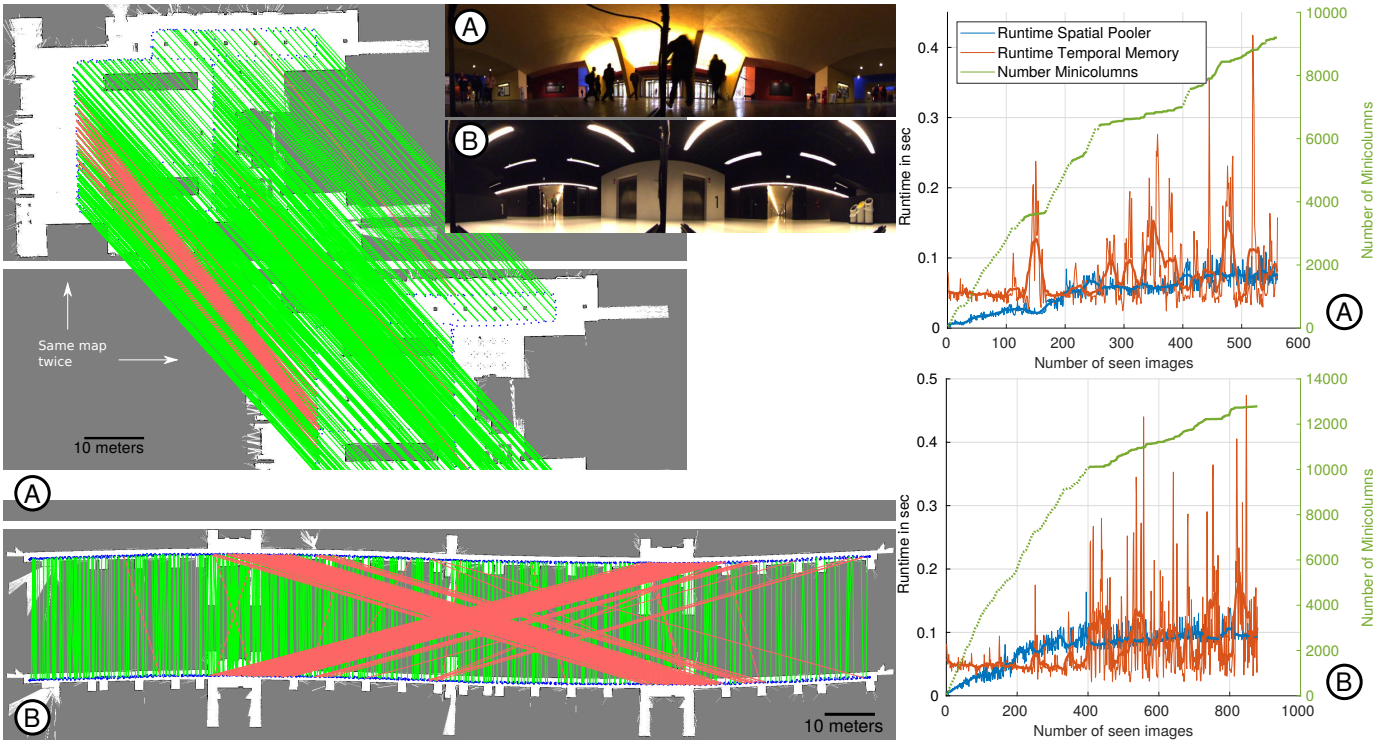

Fig. 4. Online experiments. Thick curves are 21-frame sliding averages of thin curves. Dotted green lines indicate exploration, solid lines revisits.

The rectified camera images are input for the NetVLAD+sLSBH+MCN pipeline. The output winner cell descriptors for each image are stored in a database and MCN-desciptors of new images are compared to the database using overlap metric (ratio of common ones). A fixed, beforehand chosen threshold is used to make hard binary decisions about place matchings. MCN parameters are as before, except for minicolumn activation threshold: $\theta = 0.55$. In contrast to previous experiments, we do not distinguish database and query sequences but look for matchings within a single robot trajectory. The robot is controlled remotely by hand on two trajectories A and B, illustrated in Fig. 4. Experiment A is an approx. 730m long ride with many loops through the foyer of a lecture hall building with passing students. The shown map is created from lidar measurements and lidar-based localization is used to distinguish true (shown green) and false loop closures (red, $>$ 3m position error). There are 823 true and 68 false loop closures. One area on the left side produced a group of wrong matchings and there are very few additional wrong matches. The second experiment B is an approx. 860m long ride through a 160m long corridor with a few passing people and a modern, clean architecture. Despite the very repetitive appearance, the map B in Fig. 4 illustrates the many correct loop closures due to the exploitation of sequence information. However, the two elevator areas (one is shown in the example image B) are systematically mixed up. In total, MCN achieved 618 true and 260 false loop closures.

Average time for the whole pipeline from image acquisition to relocalization decision was 2.1s. It is dominated by the image processing front-end. The curves on the right side of Fig. 4 evaluate the runtime and growth of the MCN system. The green curves show the number of minicolumns which is continuously increasing during exploration (the robot sees new areas for the first time, plotted as dotted line). The growing stops or slows down during revisits of the same areas (solid line parts). E.g., in experiment B at frame 403, the robot has traversed the corridor once in each direction and now starts recognizing places. In general, the runtime of the spatial pooler increases with increasing number of minicolumns. The runtime of temporal memory does not increase significantly over time but depends on whether the robot explores new places or reactivates predictions from known places.

VII. DISCUSSION

We extended our previous theoretical work to application on real world data. The presented minicolumn network (MCN) approach creates an internal representation that encodes sequential context. It can maintain different alternative routes through the world, is not limited to a defined sequence length (cf. parameter d_s in SeqSLAM), and can provide multiple matchings for each query image. MCN's runtime depends on the number of minicolumns. During exploration of new areas, new minicolumns are created; multiple revisits of the same area reactivate existing minicolumns. To bound the runtime during long-term operation, the usage of a fixed number of minicolumns whose connections adapt over time similar to HTM's spatial pooler in [3] should be a topic of future work. Representations in MCN are binary sparse distributed representations. Operations on these structures can be implemented very energy efficient in hardware [44] which is important for battery driven devices such as robots. The output of MCN is the sparse set of winner cells. These indices

could be used to directly address all similar places without pairwise comparison, which is essential for large datasets.

MCN takes major inspiration from Hierachical Temporal Memory, a biologically plausible theory on working principles of the human neocortex. We evaluated its properties in simulation and presented results on real world data, however, a major direction for future work is a deeper theoretical analysis (e.g., the connection to bagging ensemble classifiers [45]). Also a more in depth analysis of the capacity and runtime behavior will be beneficial for practical application. Further, there are several aspects of HTM that are interesting for future inclusion, e.g. segments, permanences, and the combination with grid-cells.

REFERENCES

[1] K. L. Ho and P. Newman, "Detecting loop closure with scene sequences," *Int. J. of Computer Vision*, vol. 74, no. 3, 2007.
[2] M. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights." in *Proc. of Int. Conf. on Robotics and Automation*, 2012.
[3] J. Hawkins, S. Ahmad, S. Purdy, and A. Lavin, "Biological and machine intelligence (bami)," 2016, initial online release 0.4. [Online]. Available: https://numenta.com/resources/biological-and-machine-intelligence/
[4] P. Neubert, S. Ahmad, and P. Protzel, "A sequence-based neuronal model for mobile robot localization," in *Proc. of KI: Advances in Artificial Intelligence*, 2018.
[5] S. Lowry, N. Sunderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *Trans. Rob.*, vol. 32, no. 1, 2016.
[6] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. of Computer Vision*, vol. 60, 2004.
[7] P. Neubert and P. Protzel, "Beyond holistic descriptors, keypoints, and fixed patches: Multiscale superpixel grids for place recognition in changing environments," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, 2016.
[8] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. of Computer Vision*, vol. 42, no. 3, 2001.
[9] N. Sünderhauf and P. Protzel, "Brief-gist - closing the loop by simple means," in *Proc. of Int. Conf. on Intelligent Robots and Systems*, 2011.
[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.
[11] N. Sünderhauf, S. Shirazi, F. Dayoub, B. Upcroft, and M. Milford, "On the performance of convnet features for place recognition," *Proc. of Int. Conf. on Intelligent Robots and Systems*, 2015.
[12] R. Arandjelovi, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," *Trans. on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, 2018.
[13] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *The Int. J. of Robotics Research*, vol. 27, no. 6, 2008.
[14] E. Johns and G. Yang, "Dynamic scene models for incremental, long-term, appearance-based localisation," in *Proc. of Int. Conf. on Robotics and Automation*, 2013.
[15] T. Naseer, L. Spinello, W. Burgard, and C. Stachniss, "Robust visual robot localization across seasons using network flows," in *Proc. of AAAI Conf. on Artificial Intelligence*, 2014.
[16] S. Lynen, M. Bosse, P. Furgale, and R. Siegwart, "Placeless place-recognition," in *Proc. of Int. Conf. on 3D Vision*, 2014.
[17] P. Hansen and B. Browning, "Visual place recognition using hmm sequence matching," in *Int. Conf. on Intel. Robots and Systems*, 2014.
[18] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, and E. Romera, "Towards life-long visual localization using an efficient matching of binary sequences from images," in *Proc. of Int. Conf. on Robotics and Automation*, 2015.
[19] O. Vysotska, T. Naseer, L. Spinello, W. Burgard, and C. Stachniss, "Efficient and effective matching of image sequences under substantial appearance changes exploiting gps priors," in *Proc. of Int. Conf. on Robotics and Automation*, 2015.
[20] O. Vysotska and C. Stachniss, "Lazy data association for image sequences matching under substantial appearance changes," in *IEEE Robotics and Automation Letters*, vol. 1, no. 1, 2016.
[21] O. Vysotska and C. Stachniss, "Relocalization under substantial appearance changes using hashing," in *Proc. of 9th Workshop on Planning, Perception, and Navigation for Intelligent Vehicles at the Int. Conf. on Intelligent Robots and Systems*, 2017.
[22] W. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1645–1661, 2013.
[23] M. Milford, G. Wyeth, and D. Prasser, "Ratslam: a hippocampal model for simultaneous localization and mapping," in *Proc. of Int. Conf. on Robotics and Automation*, 2004.
[24] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proc. of Int. Conf. on Computer Vision*, 2015.
[25] F. Walch, C. Hazirbas, L. Leal-Taix, T. Sattler, S. Hilsenbeck, and D. Cremers, "Image-based localization using lstms for structured feature correlation," in *Proc of Int. Conf. on Computer Vision*, 2017.
[26] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen, "Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization," in *Proc. of Conf. on Computer Vision and Pattern Recognition*, 2017.
[27] J. Hawkins, *On Intelligence (with S. Blakeslee)*. Times Books, 2004.
[28] J. Hawkins, S. Ahmad, and Y. Cui, "A theory of how columns in the neocortex enable learning the structure of the world," *Frontiers in Neural Circuits*, vol. 11, p. 81, 2017.
[29] "Nupic," https://github.com/numenta/nupic, accessed: 2018-05-09.
[30] J. Hawkins and S. Ahmad, "Why neurons have thousands of synapses, a theory of sequence memory in neocortex," *Frontiers in Neural Circuits*, vol. 10, p. 23, 2016.
[31] R. M. Grieves and K. J. Jeffery, "The representation of space in the brain," *Behavioural Processes*, vol. 135, pp. 113 – 131, 2017.
[32] J. Hawkins, M. Lewis, S. Purdy, M. Klukas, and S. Ahmad, "A framework for intelligence and cortical function based on grid cells in the neocortex," *Frontiers in Neural Circuits*, vol. 12, 2019.
[33] S. Ahmad and J. Hawkins, "Properties of sparse distributed representations and their application to hierarchical temporal memory," *CoRR*, vol. abs/1503.07469, 2015.
[34] I. Danihelka, G. Wayne, B. Uria, N. Kalchbrenner, and A. Graves, "Associative long short-term memory," *CoRR*, vol. abs/1602.03032, 2016. [Online]. Available: http://arxiv.org/abs/1602.03032
[35] D. Widdows and C. Trevor, "Reasoning with vectors: A continuous model for fast robust inference," *Logic Journal of the IGPL*, vol. 23, no. 2, pp. 141 – 173, 2015.
[36] P. Neubert, S. Schubert, and P. Protzel, "Learning vector symbolic architectures for reactive robot behaviours," in *Proc. of Int. Conf. on Intelligent Robots and Systems Workshop on Machine Learning Methods for High-Level Cognitive Capabilities in Robotics*, 2016.
[37] S. Purdy, "Encoding data for htm systems," *CoRR*, vol. abs/1602.05925, 2016.
[38] D. Ravichandran, P. Pantel, and E. Hovy, "Randomized algorithms and nlp: Using locality sensitive hash function for high speed noun clustering," in *Proc. of 43rd Annual Meeting on Association for Computational Linguistics*, 2005.
[39] N. Sünderhauf, P. Neubert, and P. Protzel, "Are we there yet? challenging seqslam on a 3000 km journey across all four seasons," *Proc. of Workshop on Long-Term Autonomy at the Int. Conf. on Robotics and Automation*, 2013.
[40] A. Glover, W. Maddern, M. Milford, and G. Wyeth, "FAB-MAP + RatSLAM: Appearance-based SLAM for Multiple Times of Day," in *Proc. of Int. Conf. on Robotics and Automation*, 2010.
[41] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *The Int. J. of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.
[42] H. Badino, D. Huber, and T. Kanade, "Visual topometric localization," in *Proc. of Intelligent Vehicles Symp.*, 2011.
[43] A. Glover, "Day and night with lateral pose change datasets," 2014. [Online]. Available: https://wiki.qut.edu.au/display/cyphy/Day+and+Night+with+Lateral+Pose+Change+Datasets
[44] A. Rahimi, S. Datta, D. Kleyko, E. P. Frady, B. Olshausen, P. Kanerva, and J. M. Rabaey, "High-dimensional computing as a nanoscalable paradigm," *IEEE Trans. on Circuits and Systems*, vol. 64, no. 9, 2017.
[45] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.