SMCN: Simplified mini-column network for visual place recognition

Guanyu Huang¹, An Chen*, Hongxia Gao¹, Puguang Yang¹

¹ School of Automation Science and Engineer, South China University of Technology, Guangzhou, Guangdong, 510640, China
* Corresponding author’s e-mail: chenan@scut.edu.cn

Abstract. Visual place recognition (VPR) is challenging because the places to be recognized are often affected by complex environmental changes. It is important for loop closure detection in SLAM. In recent years, a large range of approaches have been developed to address this challenge. Among these approaches, the use of biological heuristics and structural information have improved the VPR performance. In this paper, we attempt to improve the biological heuristic method MCN, and combine the available but often overlooked structural information intra-set similarity to propose our first algorithm SMCN. Furthermore, we propose a novel temporal filter that considers temporal continuity and combines it with SMCN to get our second algorithm SMCNTF. Evaluation of both algorithms on ten dataset combinations shows that, our best model SMCNTF has a maximum increase of 32% in average precision and at least a 100-fold increase in computing efficiency. Moreover, fewer parameters need to be tuned comparing with MCN. Generally, our algorithm is much better than MCN.

1. Introduction
Visual place recognition (VPR) is receiving increasing attention from communities of computer vision and robot autonomous navigation recently. Performing VPR needs to consider the various appearance of places due to uncertain factors such as day-night changes, view-point changes, and season changes, which makes VPR a challenging problem [1] [2]. The pipeline of addressing VPR is generally divided into two steps, which are called front-end and back-end [3]. The front-end mainly focuses on designing strong discriminant descriptors of images, which need to be robust to various changes of place appearance. Classic front-end descriptors include traditional hand-crafted local descriptors such as SIFT [4], ORB [5] and deep-learning based descriptors such as AlexNet [6] and NetVLAD [7]. The similarity matrix is then obtained by pairwise matching between the database descriptor and query sequence descriptor. There are lots of false positives in the best matching only by taking the maximum similarity of the pairwise similarity matrix. Therefore, the back-end method is applied to improve the VPR performance based on the front-end descriptor.

The back-end method leverages prior information of descriptors or similarity matrix to improve the performance of VPR. For example, the well-known SeqSLAM [8] applies sequential search in similarity matrix while Delta Descriptor [9] exploits change-based information in image descriptors. In recent years, biologically inspired methods arouse the attention of VPR researchers, which gives birth to work like MCN [10] [3], CANN [11], etc. It is worth noting that the structural information of the self-similarity matrix is less studied by researchers, and its combination with biological methods is even less.
In this work, inspired by MCN and considering that the self-similarity matrix is more reliable than the mutual similarity matrix, we simplify and modify MCN greatly and proposed the SMCN. In SMCN model, mini-column network structure in MCN is replaced by the intra-set candidates generated by the self-similarity matrix, and retains the core idea of MCN—the predictive effect of the locations at the previous time contributes to the prediction of the location at the current time. Furthermore, considering the temporal continuity, we design a novel temporal filter (TF). Based on a temporal loss function and self-similarity matrix, the TF solves the global optimal continuous matching and updates the final mutual similarity matrix. Combining it with the output of SMCN, we get the promotion model of SMCNTF (shown as figure 1).

The paper is divided into the following sections: Section 2 discusses the prior literature and related work; Section 3 describes the proposed approach—SMCN and temporal filter (TF); Section 4 details the experimental setup and presents the results on the benchmark datasets; Section 5 concludes the paper, highlighting potential ways to extend the current work in future. Source code is made available at https://github.com/Rick0514/VPR_SMCN.

2. Related work

2.1. Deep-learned descriptors
Deep neural network has been widely used in VPR because of its powerful ability of understanding images. For detailed descriptions of deep learning, please refer to [2]. Here is an overview of several key deep learning applications. Sünderhauf et al. thoroughly evaluated the output of each layer of the AlexNet network[12], which provides guidance for the follow-up researches. Arandjelović et al. designed an end-to-end trainable network NetVLAD for large-scale place recognition[7]. Hou et al. conducted a comprehensive investigation and experiment on the landmark-based method[13], compared 13 object proposal methods and 13 CNN features, and gave the optimal combination for place recognition. Generative adversarial networks (GANs) are also attractive in VPR. Porav et al. trained a cyclic GAN to transfer image appearance from source domain (e.g., night) to target domain (e.g., day) for image matching under adverse conditions[14], thus greatly reducing the interference of appearance differences.

2.2. Biologically inspired methods
Biologically inspired methods have received extensive attention from scholars in the field of mobile robotics in recent years. RatSLAM[15][16] is the earliest and most complete application of bio-inspired methods in SLAM. It uses the Continuous Attractor Network (CAN) to imitate the calculation model of the rodent hippocampus to construct the system's pose cell network. Because of the dynamic
characteristics of CAN, Chancán et al. used it as a temporal filter, combined with their bio-inspired front-end FlyNet\cite{11}, to achieve a good performance on place recognition. Dupeyroux et al. were inspired by ant pathfinding and designed an ant-inspired path integration method and applied it to actual robots\cite{17}. Inspired by the activity of the human cortex, Peubert et al. proposed a hierarchical temporal memory model that simulates the location of the human brain\cite{10}, and based on this, proposed the back-end method MCN related to this article\cite{3}.

2.3. Work exploiting intra-set similarity
Among back-end methods, sequence information as a very important additional structural information is widely used in the work of VPR. However, the intra-set similarity is always available but tends to be neglected. To our best knowledge, \cite{18-20} is the most similar method to our SMCN. \cite{18} uses the intra-set similarity of the database to find candidates for the query, thereby accelerating the matching efficiency. But it is a method of trading accuracy for speed. The latest work \cite{19} is very similar to \cite{18}. The innovation is that it will also use the sequence information of the database to supplement the candidates with consecutive items, and propose the algorithm to adaptively tune the parameter. \cite{20} cleverly found the inconsistency in the similarity matrix based on the intra-set similarity and resolved the conflict with the inconsistency of the self-similar matrix by modifying the similarity of the mutual similarity matrix. Unlike the methods above, based on the inspiration of MCN, ours combines the core idea of MCN with the structural information of intra-set similarity and achieves both speed and accuracy improvements.

3. Algorithmic approach

3.1. MCN overview
Hierarchical temporal memory (HTM) is a biologically plausible model for the human neocortex to process sequential information. MCN further improves the simplified HTM model and extends its application to public datasets and real-world scenarios. MCN is mainly divided into two parts: spatial pooler and temporal memory. The spatial pooler is responsible for processing the external visual input, while temporal memory, as the core part of MCN, is responsible for learning the sequence relationship of visual information. The core idea of temporal memory is that in the training process, the place visited at the previous moment and the place at this moment have a temporal and spatial adjacent relationship, and the connection between them should be considered. Therefore, in the inference process, the information that the place at the previous moment is connected to the next location at present has a guiding role (predictive role) for the decision-making of the location at this time. Our SMCN draws on this core idea and simplifies its expression. In addition to the aforementioned advantages of MCN, the following shortcomings make it necessary to improve MCN:

- Too many randomization operations are involved, resulting in huge time cost for getting stable results
- The biologically plausible expression of mini-column is redundant, resulting in long time-consumption.

3.2. Related concepts
The symbols and related concepts used in this article are shown below. We assume that the database $DB$ and the query $Q$ have the number of images $|DB|$ and $|Q|$ respectively.

**Candidates and candidate matrix:** The $k$ candidate images of a certain image $i$ in the query are the top $k$ images in the database with the highest similarity to the image $i$. Each column of the candidate matrix $C_k \in \mathbb{N}^{k \times |Q|}$ is composed of candidates for query pictures.

**Self similarity matrix and mutual similarity matrix:** self similarity matrix $S_s \in \mathbb{R}^{|DB| \times |DB|}$ refers to the similarity matrix formed by pairwise comparison of database images, and mutual similarity matrix $S_m \in \mathbb{R}^{|DB| \times |Q|}$ refers to the similarity matrix obtained by pairwise comparison of database images and query images.
**Candidate similarity matrix:** Given the groundtruth, average precision can be calculated from mutual similarity matrix given by VPR algorithm. If only the similarity of the corresponding position of the candidate matrix is retained, and the similarity of the non-candidate position is set to 0, the candidate similarity matrix \( S_c \in \mathbb{R}^{|DB| \times |Q|} \) is obtained. This is based on the fact that non-candidate pictures are often of no reference value for VPR.

### 3.3. SMCN

SMCN applies a lot of simplifications based on MCN. Considering the similarity of intra-set, the self-similarity candidates of database are used to replace the mini-columns in MCN; Index the mutual similarity matrix directly replaces the forward connection comparison in spatial pooler in MCN, which greatly saves the runtime. SMCN also includes two processes as MCN does: training and inferencing, which are detailed below.

**Training:** Training is the process of learning the connection between self-similarity candidates. As shown in figure 2(a), at \( t \), we connect the candidates at \( t - 1 \) with the candidates at present, but if it points to itself, the connection shouldn’t be established. The intention is that if the candidates at \( t - 1 \) are given, they can infer the image appearance of the place at the next moment from the images they connect to, which is the core idea of MCN. At \( t + 1 \), we continue to connect the candidate at \( t \) to the candidate at the current time, if the connection is not established yet. It is worth noting that we use the self-similarity candidate matrix to train, which takes advantage of the fact that the similarity of intra-set is more reliable than that of inter-set.

**Inferencing:** The process of inferencing is to make decisions according to visual similarity and prediction. It should be noted that the inference process is performed based on the inter-set candidates. As shown in figure 2(b), there is no predicted candidate from the previous moment \( t - 1 \), so all candidates (No. 4, 3, 0) are regarded as final candidates and marked as light blue. At \( t \), No. 4 predicts No. 6 while No. 3 predicts No.3, and No. 6 and No. 3 happen to be in the inter-set candidate matrix at \( t \), so No. 6 and No. 3 are the final candidates at \( t \). At \( t + 1 \), No. 6 predicts No. 8 while No.3 predicts No. 7, and they are in the inter-set candidate matrix, so No. 8 and No. 7 are the final candidates at \( t + 1 \) and so on.

### 3.4. Temporal filter

The similarity of the candidate similarity matrix obtained by the SMCN algorithm is taken from the mutual similarity matrix. Although a large number of false positives are excluded, there is no further improvement. In this paper, we propose a novel temporal filter to update the candidate similarity based on the temporal sequence relationship.

From a global perspective, there must be an optimal matching sequence, so that the matched image index is continuously changing (the case of loop closure will be taken into account). Therefore, we can find this optimal matching sequence first, and then use this sequence as a reference. If other candidate
matches are close to the reference, the similarity of this candidate is high. In this way, we modify the candidate similarity matrix.

Figure 3. Schematic diagram of the optimal matching sequence. Orange dots and lines represent the optimal matching sequence.

We regard the optimal matching problem as solving a special shortest path problem (see figure 3). This optimal matching sequence needs to satisfy that appearance changes of matched images should be the most continuous and stable, and that sudden changes should be avoided as much as possible. Therefore, we design the loss function as (1). For ease of understanding, only the minimum loss calculation from \( t - 1 \) to \( t + 1 \) is shown in the figure 3, and the overall situation can be deduced by analogy.

\[
\text{Loss}_{t+1} = \text{Loss}_m + \text{Loss}_s
\]
\[
\text{Loss}_m = S_m(I_{k-1}^t, t - 1) + S_m(I_1^t, t) + S_m(I_{2+1}^t, t + 1)
\]
\[
\text{Loss}_s = S_s(I_{k-1}^t, I_1^t) + S_s(I_1^t, I_{2+1}^t)
\]

In the formula (1), \( \text{Loss}_m \) is called "candidate loss", which measures the loss of visually distinguishing the database and query images; \( \text{Loss}_s \) is called "temporal loss", which uses the self-similarity matrix \( S_s \). If the appearance difference between the two matched images is large, the loss will be relatively large. With the loss function, we can use dynamic programming algorithm to solve the "shortest path" problem.

After obtaining the optimal matching sequence, we start to update the candidate similarity matrix \( S_c \). We assume that the optimal matching sequence has the maximum similarity. If the difference between the candidate index \( I \) and the optimal matching index \( I_{\text{best}} \) is less than a threshold \( \phi \), we consider this candidate index and the optimal matching index have the same similarity, otherwise their similarity is taken from the self-similarity matrix \( S_s \).

\[
S_c(I) = \begin{cases} 
1, & |I - I_{\text{best}}| < \phi \\
S_s(I, I_{\text{best}}), & \text{otherwise} 
\end{cases}
\]

4. Experimental evaluation

4.1. Experimental setup

In the following experiments, AlexNet and NetVLAD are used as visual front-end. For AlexNet, we use the official implementation provided by Pytorch and load the pre-trained model trained on ImageNet[21]. We take the output of its flattened 43264-dimensional conv4-layer as the descriptor. For NetVLAD, we also use the Pytorch version of the code implementation and load the model trained on the Pittsburgh 250K dataset[22]. We take its 32768-dimensional output as the descriptor. Cosine similarity is used to evaluate the similarity of two descriptors. For evaluation metric, we use average precision computed as the area under the precision-recall curve (using trapezoidal integration)[3].
4.2. Datasets

Algorithm 1: SMCN

Input: final candidate number k, intra-set candidate number k1, inter-set candidate number k2, Sn, Sm
Output: Ck, Sn

1. \( C_{k,1} = \text{getTopk}(S_n, k) \), Con = \{ Coni = \emptyset | 1 \leq i \leq |DB| \}
2. //Training
3. for \( 1 \leq i \leq |DB| - 1 \) do
4. for \( j \in C_{k,1} \) do
5. \( \text{Con}_{j} = \text{Con}_{j} \cup C_{k,1,j+1} \)
6. \( C_{k,2} = \text{getTopk}(S_m, k) \), \( \text{preCan} = C_{k,2,1} \)
7. \( C_{k,3} = \{ C_{k,3} = \emptyset | 1 \leq i \leq m \}, C_{k,3} = \text{getTopk}(C_{k,2,1}, k) \)
8. //Inferencing
9. for \( 2 \leq i \leq |Q| \) do
10. \( \text{tmp} = \text{findAllPredictions} \text{(preCan)} \)
11. \( \text{pCan} = \text{find}(C_{k,2,j}, \text{tmp}) \)
12. if \( |\text{pCan}| > 0 \) then
13. \( C_{k,j} = C_{k,j} \cup \text{getTopk}(\text{pCan}, k) \), \( \text{preCan} = C_{k,j} \)
else
14. \( C_{k,j} = C_{k,j} \cup \text{getTopk}(C_{k,2,j}, k) \)
16. \( S_n = \text{getSimilarity}(C_{k,j}) \)

(a) SMCN

Algorithm 2: Temporal Filter(TF)

Input: threshold \( \Phi \), Ck, Sm, Sn
Output: modified similarity matrix \( S_m^{'} \)
1. \( \text{loss} = \text{cosSim}(C_k) \) //replace candidate indices with corresponding similarity.
2. \( \text{lastCan} = C_{k,1} \)
3. // dynamic programming
4. for \( 2 \leq i \leq |Q| \) do
5. for \( 1 \leq j \leq |C_{k,j}| \) do
6. \( \text{lossSet} = \emptyset \)
7. for \( p \in \text{lastCan} \) do
8. \( \text{lossSet} = \text{lossSet} \cup \text{getLoss}(\text{loss}, C_{k,j}, p) \)
9. \( \text{loss}_{j} = \text{min}(\text{lossSet}) \)
10. \( \text{bestSequence} = \text{backTracking}(\text{loss}) \)
11. // modify similarity matrix
12. for \( 1 \leq i \leq |Q| \) do
13. for \( j \in C_{k,j} \) do
14. if \( |j - \text{bestSequence}| > \Phi \) then
15. \( S_{m,j,i} = S_{m,j,bestSequence} \)
else
16. \( S_{m,j,i} = 1 \)

(b) SMCNTF

The following experiments are based on three datasets with different characteristics regarding appearance changes and viewpoint changes. Nordland[23]: Time and viewpoint synchronized rides along a single train track once in each season. In the experiment, we used the down-sampled (224×224) images in the first test set provided by [24](1150 pictures per season) and assume that the locations represented by the 9 adjacent images of each image are the same place. Gardens Point Walking[25]: Hand-held camera on a single route on campus, two times at day and once at night with controlled viewpoint deviations. The images are down-sampled(224x224). We assume that the locations represented by the 3 adjacent images of each image are the same place. Oxford RobotCar[26]: Recorded with a car equipped with several cameras and lidars over a period of over a year. We used the images collected by the center camera of the trinocular camera of the first part of the journey(day, night, snow) corresponding to 2014-12-16-18-44-24_s_tereo_centre_01, 2015-02-03-08-45-10_s_tereo_centre_01, 2015-05-22-11-14-30_s_tereo_centre_01). Adopted the method of [27], with the assistance of GPS, the 6001 images in a traversal were sampled every 2m, and finally, 400 to 700 images were obtained per traversal. GPS is used as groundtruth. The images are down-sampled(224x224) as well. Places within a radius of 5m are considered the same place.

4.3. Instability of MCN

In Sec. 3.1, we mentioned the instability of MCN. Here is the experimental demonstration. NetVLAD descriptor is used and the dataset is GardensPoint(day_right vs. night_right). The parameters of MCN are the same as those in [3], and the parameters of SMCNTF are \( k = 5, k_1 = 2, k_2 = 20 \). By changing the number of input connections of each mini-column and running it five times, observe fluctuation of the performance(see figure 5).

As dimension of input connections increases, the time consumption of MCN increases linearly. From AP, there is deviation each of the five runs. The largest deviation occurs when the dimension is 100, reaching 3.6%. In addition, increasing the dimension does not necessarily guarantee the reduction of the deviation. Compared with our best method SMCNTF proposed in this paper, SMCNTF is at least 177 times faster. From AP, SMCNTF is 16.4% higher than the best result in MCN. Moreover,
SMCN is stable. As long as the three hyperparameters are determined, there is no probability of fluctuation.

Figure 5. Experimental simulation of MCN instability.

4.4. Hyperparameters of SMCN

Three hyperparameters should be determined when using SMCN: the number of final candidates $k$, the number of intra-set candidates $k_1$, and the number of inter-set candidates $k_2$. Empirically, we recommend $k = 0.02n$ to ensure enough candidates and filter out most of false positives. In the following experiment, we discuss the selection and significance of the remaining parameters ($k_1, k_2$). We evaluate on three dataset combinations (summer vs. winter from Nordland, day_left vs. night_right from Gardens Point, and snow vs. night from Oxford Robocar). The variable is $\{(k_1, k_2) \mid k_1 \in \{2, 3, 4, 5\}, \quad k_2 \in \{0.03n, 0.05n, 0.07n, 0.09n\}\}$, and the effect on the performance is observed.

The number of intra-set candidates $k_1$ controls the number of learning connections. The smaller $k_1$ is, the smaller number of predictions given by each location but the higher credibility. However, a too small $k_1$ will affect the recall. The number of inter-set candidates $k_2$ controls the number of candidates retained during inferencing. With a small $k_2$, only a small part of the candidates with high visual similarity will be retained, but the predicted candidates may be missed, so $k_2$ should not be too small. It can be inferred from figure 5 that the parameter configuration is more sensitive to the dataset, but not to the descriptor. For Nordland and Gardens Point, the parameter $(4, 0.03n)$ is the best overall. For Oxford Robotcar, the parameter $(2, 0.05n)$ will have better performance. In fact, there is a basis for the parameters selection. From the perspective of AP, the front-end descriptors on Nordland and Gardens Point perform relatively well, so $k_1$ can be appropriately larger, and $k_2$ can be appropriately reduced. For Oxford Robotcar, the front-end descriptor discrimination is relatively poor, so it is necessary to choose a smaller $k_1$ to make the model more cautious, and a larger $k_2$ is chosen since a larger proportion of candidates are retained from inter-set. It is helpful for SMCN to select the correct match. In the case of poor performance of the front-end descriptor, the candidates from inter-set are unreliable, and $k_2$ needs to be increased to include more correct candidates.

4.5. VPR performance

From the perspective of AP, the performance of MCN is flat or even degraded compared to pairwise, while SMCN does not. The worst case (snow vs. night with NetVLAD) is increased by 2%, and the best situation (spring vs. winter with AlexNet) increased by 29%. Comparing with MCN, SMCN reduces by 5% in the worst case (day_left vs. day_right with NetVLAD), but improves in most cases (16 out of 20), and in the best case (spring vs. winter with AlexNet) 25%. SM CNNT further improves VPR performance based on SMCN. The former surpasses the latter in all cases, and the worst case is
also increased by 3%. Although it is not as good as MCN in one test (day_left vs. day_right with NetVLAD), the other cases showed the best performance.

Figure 6. Evaluation on hyper parameters $k_1$, $k_2$. The red columns are obtained using AlexNet descriptor, and the green ones are obtained using NetVLAD descriptor.

Table 1. Experimental results for methods proposed with Alexnet or NetVLAD front-end. Metrics include average precision and runtime. Best values per row are bold.

| Dataset  | Database | Query          | Pairwise AP | MCN AP | MCN Runtime/s | AlexNet SMCN(ours) AP | SMCNTF(ours) AP | NetVLAD AP |
|----------|----------|----------------|-------------|--------|--------------|-----------------------|-----------------|-----------|
| Nordland | spring   | winter         | 0.24        | 0.28   | 1441.277     | 0.53                  | 3.031           | 0.60      | 4.101    |
|          | summer   | winter         | 0.20        | 0.21   | 1372.501     | 0.43                  | 3.023           | 0.53      | 4.386    |
|          | fall     | winter         | 0.23        | 0.26   | 1331.676     | 0.40                  | 3.050           | 0.49      | 4.728    |
|          | spring   | fall           | 0.36        | 0.39   | 1303.696     | 0.54                  | 3.044           | 0.57      | 4.544    |
| Gardens Point | day_left    | day_right      | 0.44        | 0.49   | 54.718       | 0.55                  | 0.190           | 0.59      | 0.219    |
|          | day_right | night_right    | 0.43        | 0.48   | 35.756       | 0.61                  | 0.190           | 0.65      | 0.203    |
|          | day_left  | night_right    | 0.21        | 0.20   | 36.662       | 0.31                  | 0.191           | 0.34      | 0.213    |
| Oxford Robotcar | day   | night          | 0.12        | 0.17   | 82.687       | 0.19                  | 0.635           | 0.29      | 0.780    |
|          | day      | snow           | 0.29        | 0.37   | 73.714       | 0.37                  | 0.607           | 0.47      | 0.739    |
|          | snow     | night          | 0.09        | 0.10   | 162.619      | 0.15                  | 0.913           | 0.26      | 1.172    |
| Nordland | spring   | winter         | 0.09        | 0.11   | 1433.413     | 0.20                  | 2.405           | 0.27      | 3.751    |
|          | summer   | winter         | 0.09        | 0.10   | 1372.111     | 0.21                  | 2.315           | 0.27      | 3.444    |
|          | fall     | winter         | 0.10        | 0.13   | 1436.419     | 0.24                  | 2.328           | 0.32      | 3.631    |
|          | spring   | fall           | 0.17        | 0.21   | 1415.576     | 0.30                  | 2.394           | 0.39      | 4.098    |
| Gardens Point | day_left    | day_right      | 0.59        | 0.73   | 31.137       | 0.68                  | 0.140           | 0.72      | 0.154    |
|          | day_right | night_right    | 0.45        | 0.54   | 30.572       | 0.58                  | 0.130           | 0.62      | 0.147    |
|          | day_left  | night_right    | 0.31        | 0.37   | 31.388       | 0.42                  | 0.133           | 0.48      | 0.148    |
| Oxford Robotcar | day   | night          | 0.07        | 0.09   | 72.226       | 0.08                  | 0.506           | 0.13      | 0.636    |
|          | day      | snow           | 0.33        | 0.40   | 89.735       | 0.38                  | 0.514           | 0.46      | 0.663    |
|          | snow     | night          | 0.05        | 0.05   | 127.393      | 0.07                  | 0.758           | 0.12      | 0.993    |

Our experiment ran on a cloud server with an Intel Xeon CPU with 4 cores and 8GB RAM. Compared with MCN, the runtime consumed by the method proposed in this paper is greatly reduced. For the Nordland dataset, the runtime of MCN is more than 500 times that of SMCN and more than 300 times that of SMCNTF. For Gardens Point, the runtime of MCN is more than 200 times that of SMCN and SMCNTF. For Oxford Robotcar, the runtime of MCN is more than 100 times that of
SMCN and SMCNTF. When the dataset images are less than 1000, the runtime of SMCN and SMCNTF is not much different, so inserting TF after SMCN is recommended.

5. Conclusions
Many excellent bio-inspired approaches have emerged in visual place recognition in recent years, showing us the great potential of bio-inspired methods. In this paper, we propose SMCN based on the biologically inspired MCN and exploiting intra-set similarity. Furthermore, a novel temporal filter is proposed considering temporal continuity combined with SMCN to obtain SMCNTF. We have demonstrated that our proposed algorithm is superior to MCN in terms of average precision and computing efficiency through a series of experiments. Moreover, our algorithm needs to adjust only three hyperparameters, which is very convenient for migration to another scene.

Future work will focus on the refinement and migration of the algorithm. Firstly, the training process of SMCN can integrate more biological mechanisms to further improve performance. Secondly, auto-tuning of hyperparameters is also an important topic, which requires a deeper understanding of its impact on VPR performance. Finally, the proposed temporal filter can be integrated with state-of-the-art visual place recognition methods proposed currently, and it is possible to achieve a further performance improvement.

Acknowledgments
This research was supported by the Nature Science Foundation of Guangdong Province, China (Grant No. 2019A1515011041), Xijiang Innovation Team Project and National Natural Science Foundation of China (61873096, 62073145).

References
[1] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford. (2015) Visual place recognition: A survey. IEEE Transactions on Robotics, vol. 32, no. 1, pp. 1–19.
[2] X. Zhang, L. Wang, and Y. Su. (2021) Visual place recognition: A survey from deep learning perspective. Pattern Recognition, vol. 113, p. 107760.
[3] P. Neubert, S. Schubert, and P. Protzel. (2019) A neurologically inspired sequence processing model for mobile robot place recognition. IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 3200–3207.
[4] D. G. Lowe. (2004) Distinctive image features from scale-invariant keypoints. International journal of computer vision, vol. 60, no. 2, pp. 91–110.
[5] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. (2011) Orb: An efficient alternative to sift or surf. 2011 International conference on computer vision. Ieee, pp. 2564–2571.
[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. (2012) Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, vol. 25, pp. 1097–1105.
[7] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. (2016) Netvlad: Cnn architecture for weakly supervised place recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5297–5307.
[8] M. J. Milford and G. F. Wyeth. (2012) Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. IEEE international conference on robotics and automation. IEEE, pp. 1643–1649.
[9] S. Garg, B. Harwood, G. Anand, and M. Milford. (2020) Delta descriptors: Change-based place representation for robust visual localization. IEEE Robotics and Automation Letters, vol. 5, no. 4, pp. 5120–5127.
[10] P. Neubert, S. Ahmad, and P. Protzel. (2018) A sequence-based neuronal model for mobile robot localization. in Joint German/Austrian Conference on Artificial Intelligence (Kunstliche Intelligenz). Springer, pp. 117–130.
[11] M. Chancan, L. Hernandez-Nunez, A. Narendra, A. B. Barron, and M. Milford. (2020) A hybrid compact neural architecture for visual place recognition. IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 993–1000.
[12] N. Sünderhauf, S. Shirazi, F. Dayoub, B. Upcroft, and M. Milford. (2015) On the performance of convnet features for place recognition. IEEE/RSJ international conference on intelligent robots and systems(IROS). IEEE, pp. 4297–4304.
[13] Y. Hou, H. Zhang, and S. Zhou. (2018) Evaluation of object proposals and convnet features for landmark-based visual place recognition. Journal of Intelligent & Robotic Systems, vol. 92, no. 3, pp. 505–520.
[14] H. Porav, W. Maddern, and P. Newman. (2018) Adversarial training for adverse conditions: Robust metric localisation using appearance transfer. IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 1011–1018.
[15] M. J. Milford, G. F. Wyeth, and D. Prasser. (2004) Ratslam: a hippocampal model for simultaneous localization and mapping. IEEE International Conference on Robotics and Automation, Proceedings ICRA, vol. 1. IEEE, pp. 403–408.
[16] M. J. Milford and G. F. Wyeth. (2008) Mapping a suburb with a single camera using a biologically inspired slam system. IEEE Transactions on Robotics, vol. 24, no. 5, pp. 1038–1053.
[17] J. Dupeyroux, J. R. Serres, and S. Viollet. (2019) Antbot: A six-legged walking robot able to home like desert ants in outdoor environments. Science Robotics, vol. 4, no. 27.
[18] P. Neubert, S. Schubert, and P. Protzel. (2015) Exploiting intra database similarities for selection of place recognition candidates in changing environments. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Workshop on Visual Place Recognition in Changing Environments.
[19] S. Schubert, P. Neubert, and P. Protzel. (2021) Beyond ann: Exploiting structural knowledge for efficient place recognition. arXiv preprint arXiv:2103.08366.
[20] P. Neubert, S. Schubert, and P. Protzel. (2021) Resolving place recognition inconsistencies using intra-set similarities. IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 2084–2090.
[21] P. Team. Pytorch implementation of alexnet. [Online]. Available: https://pytorch.org/hub/pytorch_vision_alexnet/
[22] N. van Noord. Pytorch implementation of netvlad. [Online]. Available: https://pytorch.org/hub/pytorch_vision_netvlad/
[23] N. Sünderhauf, P. Neubert, and P. Protzel. (2013) Are we there yet? challenging seqslam on a 3000 km journey across all four seasons. in Proc. of Workshop on Long-Term Autonomy, IEEE International Conference on Robotics and Automation (ICRA).
[24] D. Olid, J. M. Facil, and J. Civera. (2018) Single-view place recognition under seasonal changes. arXiv preprint arXiv:1808.06516.
[25] A. Glover. (2014) Day and night with lateral pose change datasets. [Online]. Available: https://wiki.qut.edu.au/display/cyphy/Day+and+Night+with+Lateral+Pose+Change+Dataset
[26] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. (2017) 1 year, 1000 km: The oxford robotcar dataset. The International Journal of Robotics Research, vol. 36, no. 1, pp. 3–15.
[27] S. Garg, N. Sünderhauf, and M. Milford. (2018) Lost? appearance-invariant place recognition for opposite viewpoints using visual semantics. arXiv preprint arXiv:1804.05526.