Self-Supervised Place Recognition by Refining Temporal and Featural Pseudo Labels From Panoramic Data

Chao Chen ©, Zegang Cheng ©, Xinhao Liu ©, Graduate Student Member, IEEE, Yiming Li ©, Li Ding ©, Ruoyu Wang ©, and Chen Feng ©, Member, IEEE

Abstract—Visual place recognition (VPR) using deep networks has achieved state-of-the-art performance. However, most of them require a training set with ground truth sensor poses to obtain positive and negative samples of each observation’s spatial neighborhood for supervised learning. When such information is unavailable, temporal neighborhoods from a sequentially collected data stream could be exploited for self-supervised training, although we find its performance suboptimal. Inspired by noisy label learning, we propose a novel self-supervised framework named TF-VPR that uses temporal neighborhoods and learnable feature neighborhoods to discover unknown spatial neighborhoods. Our method follows an iterative training paradigm which alternates between: (1) representation learning with data augmentation, (2) positive set expansion to include the current feature space neighbors, and (3) positive set contraction via geometric verification. We conduct auto-labeling and generalization tests on both simulated and real datasets, with either RGB images or point clouds as inputs. The results show that our method outperforms self-supervised baselines in recall rate, robustness, and heading diversity, a novel metric we propose for VPR.

Index Terms—Convolutional neural network, feature maps, global representation, image descriptors, place recognition, point cloud, retrieval results, robust representation, street view, self-supervised task, self-supervised learning, visual localization, visual place recognition, viewpoint changes.

I. INTRODUCTION

VISUAL place recognition (VPR), which aims to identify previously visited places based on current visual observation, is a well-known problem in computer vision and plays a crucial role in autonomous robots. Meanwhile, VPR is closely related to re-localization, loop closure detection, and image retrieval. Despite all efforts, VPR remains a difficult task due to various challenges such as perceptual aliasing and view direction differences [1], [2]. Classic VPR methods based on hand-crafted feature matching do not require supervised learning, but are less robust to the challenges mentioned above [3], [4]. Thus, learning-based methods have been proposed to learn local or global feature descriptors [5] by place classification [6] or contrastive-like similarity learning [7]. Some works also use a sequence of images instead of a single image to mitigate perceptual aliasing issues. [8], [9].

So far, most learning-based VPR methods are supervised, focusing on either learning better feature representations or designing robust matching strategies. They assume that ground truth camera poses are available in their training sets, for obtaining the positive and negative training triplets of each visual observation [5], [7], or defining place categories [6]. Although noisy ground truth, such as inaccurate GPS signal, are available in outdoor environments, obtaining such labels is rather non-trivial for indoor scenes. One possible approach is to use existing visual SLAM or SM methods to estimate camera poses. However, these methods can not always guarantee accurate pose estimations due to various challenges including the need for robust loop closing from VPR itself. Considering human’s extraordinary VPR ability that does not seem to need ground truth pose information for its training, we ask the following question: is it possible to relax such an assumption and design a learning-based VPR approach without pose-dependent supervision?

To achieve this goal, our main idea is to leverage fixed temporal neighborhoods and learnable feature neighborhoods to discover the unknown spatial neighborhoods (which require ground truth poses to compute), leading to a self-supervised VPR method shown in Fig. 1. We are inspired by research work utilizing sensory streams (RGB videos or point cloud sequences) to obtain the positive and negative neighbors in the temporal domain such as [10]. However, we find that VPR learned from temporal cues alone will miss spatial neighboring places with large viewpoint differences, because temporal neighbors tend to share similar viewpoints.

This is suboptimal in applications such as visual navigation or loop closure for SLAM. To automatically discover the true spatial neighbors with diverse viewpoints, we propose a novel iterative learning strategy inspired by noisy label training such as bootstrapping [11]. More specifically, we exploit the temporal information for label initialization as shown in Fig. 1(a). Afterward, a feature representation will be learned based on the selected triplets. Then, as shown in Fig. 1(b), we select the training triplets using the feature space by: (1) adding feature-space neighbors as tentative positives, (2) rejecting false positives and false negatives via geometric verification, in order to further
refine the feature representation. Note that the above steps are iteratively conducted until convergence.

To evaluate our self-supervised VPR with Temporal and Feature neighborhood interactions (TF-VPR), we generate a simulated RGB dataset [12], and apply a real-world point cloud dataset KITTI-360 [13]. All the datasets are sequentially-collected sensory streams. Meanwhile, we develop a novel metric to measure the heading diversity of the retrieval results. In summary, our contributions are:

1) We propose a novel self-supervised VPR framework termed TF-VPR that eliminates pose-dependent supervision by iteratively refining pseudo labels from temporal and feature neighborhoods.
2) We propose a new evaluation metric to assess the heading diversity of VPR retrieval results.
3) We conduct comprehensive experiments using both RGB and point cloud datasets to show the auto-labeling and generalization capabilities of TF-VPR in comparison with other baseline methods.

II. RELATED WORK

Visual place recognition: Visual place recognition (VPR) is the problem of identifying a previously visited place based on visual information [1]. Existing Visual Place Recognition (VPR) methods mainly lie in two categories: (1) VPR techniques based on local features [14], [15], [16], [17], and (2) state-of-the-art VPR techniques using deep learning [7], [18], [19], [20], [21].

Within category (1), VPR techniques based on local features can be further subdivided according to the type of local features they utilize: hand-crafted or learnable. Hand-crafted approaches extract key points and descriptors manually, relying on local structures, gradient orientations, or intensity patterns in the image [14], [15]. In contrast, learnable feature approaches leverage machine learning algorithms to extract features, allowing for more flexible and adaptive representations [16], [17]. However, challenges such as perceptual aliasing and variations in view direction persist in this category [1], [2]. Within category (2), NetVLAD [7] is a seminal deep-learning-based VPR framework, followed by various research extensions such as learning powerful feature representation [22], [23], designing robust matching strategies [8], [9], [24], and investigating different input modalities in VPR [25], [26]. However, most methods are supervised by pose-dependent data [7], [18], [19]. To relax such a constraint, several attempts have been made [27], [28], [29], yet failed to handle diverse re-visiting viewpoints. The most relevant works to our method are VPR calibration [30] and the semi-parametric topological memory (SPTM) [10]. To avoid direct supervision from GPS, the former approach [30] tries to obtain weak VPR supervision from SLAM. However, the quality of such supervision strongly depends on having good VPR for loop closing in SLAM in the first place, not to mention other challenges in SLAM. In contrast, the latter [10] utilizes temporal positives and negatives to train a binary classification network for adding edges for topological mapping, similar to VPR. However, since temporal neighbors tend to have very similar viewpoints, SPTM still struggles to recognize revisits of the same place from different viewpoints. To the best of our knowledge, no research has addressed self-supervised VPR that can recognize places observed from various viewpoints as either 2D images or 3D point clouds.

Noisy label learning: Noisy labels become a problem as training data size increases, resulting in degraded performance [31].

To mitigate the issues of noisy labels, several learning attempts [32] have been made from directions including latent variable optimization [33], loss function design [34], consistency regularization [35], [36], and pseudo-label-based self-training [11], [37], [38], [39].

Among all pseudo-label methods, label refurbishment was first introduced by Bootstrapping [11]. Later on, another method addressed this problem using a self-training-based approach with an iterative workflow [38] which was used as a baseline architecture in their more recent framework confidence regularized [39]. This iterative approach has also been used in keypoint matching [40]. Our iterative training paradigm is similar at a high level but needs to address unique challenges and opportunities in obtaining image-level description for robust retrieval in VPR, unlike pixel-level segmentation [38], [39] or keypoint-level descriptor matching [40] that cannot leverage temporal information.

Contrastive learning: It is a self-supervised learning technique to find feature representations that differentiate similar data pairs from dissimilar ones without labels. Data augmentation is often used, and the learning objective is to decrease the feature distances between the original and augmented images (positive samples), while increasing those distances between different images (negative samples) [41], [42], [43]. In VPR, NetVLAD [25] uses the triplet loss which is similar to contrastive learning, yet relies on ground truth pose to define positive/negative samples.

VPR evaluation: There are several evaluation metrics for visual place recognition, e.g., the popular AUC-PR [44] provides a good overview of precision and recall performance but is less indicative in the cases when ground truth match could take multiple values. Recall Rate@N, as used in [7], [45], [46], is designed to address such cases that the correct retrieval may be in the top-N results, and multiple correct query matches are neither penalized nor rewarded. However, existing VPR metrics rarely evaluate the viewpoint diversity of the retrieved results [1], which is important in downstream applications such as SLAM. In this work, we develop such a metric to fill this gap. It assesses a VPR model’s capacity to recognize places revisited from different directions.

III. METHOD

A. Problem Setup and Formulation

Consider a mobile agent (e.g., a robot, a self-driving car) equipped with a panoramic RGB camera or a 360° LiDAR
sensor while moving in an environment (typically in GPS-denied regions) without robust localization for accurate positions and orientations. At each step $i$ of its movement, the agent takes one observation $o_i$. For cameras, $o_i \in \mathbb{R}^{H \times W \times 3}$ is an RGB image. For LiDARs, $o_i = \{p_i^j\}_{j=1}^{N_i}$ is a 3D point cloud, where $p_i^j \in \mathbb{R}^3$ is the $j$-th point in the point cloud, and $N_i$ denotes the number of points.

To obtain a good spatial understanding of the environment, the agent needs visual or LiDAR place recognition to recognize revisits of places. Nowadays this is typically done by training a neural network $f$ with learnable parameters $\theta$ that can extract a global feature vector $f_\theta(o_i)$ for each observation $o_i$ in a training dataset, such that each observation’s feature space neighbors are also its spatial neighbors. Our approach to addressing the task involves two primary modes: (1) generalization and (2) auto-labeling.

In generalization mode, the evaluation criteria focus on the model’s capacity to extend beyond the observed data to make predictions on previously unseen instances. The dataset $D$ consists of two subsets: the training set $D_{\text{train}}$ and the test set $D_{\text{test}}$. During the training phase, the model $f_\theta$ is trained on a tuple $\langle q, p^q, n^q \rangle$, where the query $q$, its positives $p^q$, and its negatives $n^q$ are all drawn from the training set $D_{\text{train}}$. Subsequently, during the testing phase, we partition the testing set $D_{\text{test}}$ into the query set $Q$ and the database set $D_{\text{base}}$. The trained model $f_\theta$ is tested on all frames in $Q$ and $D_{\text{base}}$ to derive their feature vectors. Following this, for each query $q \in Q$, its top $k$ neighbors are retrieved from the database $D_{\text{base}}$ based on the euclidean feature distance $d_{Q,j}^i(i,j)$ between the query and the database. These neighbors are compared with the ground truth to evaluate the model’s cross-domain performance during inference.

The auto-labeling mode’s setup is mostly similar to generalization, except that auto-labeling mode does not require a testing phase. Auto-labeling aims to generate a binary topology graph for a dataset $D$. Each node in this graph represents an observation in $D$, and an edge signifies that two observations are taken in proximity. Unlike the generalization mode, we assume the ground truth of this graph is not available during auto-labeling (but only during the evaluation). In auto-labeling, we optimize the model $f_\theta$ on dataset $D$. Subsequently, a binary graph is generated based on the feature vectors extracted from $f_\theta$. An edge between two images can be connected if the feature distance $d_{Q,j}^i(i,j)$ between them is below a threshold. Thus, auto-labeling is a self-supervised overfitting on the dataset $D$. This process can generate topological graphs for SLAM, such as in DeepMapping2 [47].

Regardless of modes, the core idea of TF-VPR is an iterative process of noisy label learning and refinement, designed to train a VPR network without relying on ground truth spatial neighborhoods for direct or weak supervision. This is achieved by utilizing the relationship between temporal, spatial, and feature neighborhoods. In Fig. 2, the pipeline consists of four stages discussed in the following subsections: (1) labeling, (2) training, (3) expansion, and (4) contraction.

### B. Initial Data Labeling and Data Augmentation

Inspired from [7], [10], the temporal adjacent neighbor should be also spatially adjacent. In Fig. 3 on the left, given a training query $q_i$, we generate temporal positive set $P_{q_i}$ and negative set $N_{q_i}$ based on the temporal index $i$.

$$o_j \in P_{q_i}^t \iff |i-j| < n,$$  \hspace{1cm} (1)

$$o_j \in N_{q_i}^t \iff |i-j| > kn.$$  \hspace{1cm} (2)

For a query $q_i$, we select $n$ temporal neighbors ahead and behind it as its temporal positives, as indicated in (1). Similarly, in (2), its temporal negatives are defined as observations that are at least $kn$ time steps away. The hyperparameter $k$ in (2) helps to adjust the boundary for the temporal negatives.

Before feeding the data into the network for training, the augmentation module performs a random horizontal rolling on
panoramic image input or a random rotation on point cloud input. The augmentation step simulates a random rotation of the sensor heading, allowing the network to obtain new observations from different orientations at the same location. This is critical if the same location is visited multiple times from different viewpoints. Following forward propagation of the network, we compute the loss and perform back propagation using the pseudo label from the first stage.

Discussion on the initial labels: The current setup may contain noisy labels, but in practice, their proportion is very low. This is because empirically the ratio of positives to negatives for each query is low, making the probability of sampling spatial positives from temporal negatives very low.

C. Expansion

After Section III-B, expanding the neighborhoods is necessary because the current positive set \( P_q \), can be limited in size and diversity. Additionally, the negative set \( N_q \) may be noisy since temporally distant observations could still be spatial neighbors due to place revisits, as shown in Fig. 3.

Between epochs, our iterative workflow expands neighborhoods by evaluating the training dataset, as the features from the network encode the spatial information. For each query \( q_i \), we apply KNN on feature space to retrieve the top \( k \) feature neighbors as potential positives \( \hat{P}_q \). The number of retrieved neighbors is determined dynamically since not all queries have the same number of positives. Specifically, we first compute the minimum feature distance \( d_{f_o}^2 \) between each query \( q_i \) and its temporal neighbors \( P_{o_i} \) as threshold

\[
\tau_i = \min_{o_i \in P_{o_i}} \left( d_{f_o}^2 (q_i, o_i) \right). \tag{3}
\]

Next, for the sake of computation speed, we only focus on finding the positive candidates from the top \( k \) nearest feature neighbors using KNN. An observation \( o_j \) in these feature neighbors, with a smaller feature distance than the threshold \( \tau_i \), will be selected as a potential candidate

\[
\hat{P}_{q_i} = \{ o_j : d_{f_o}^2 (q_i, o_j) < \tau_i \}. \tag{4}
\]

Existing VPR training methods typically prioritize hard negative mining over positive mining, both utilizing the feature space to refine the training dataset. However, in Section IV, we emphasize the crucial role of positive mining. It reduces the performance gap between self-supervised and supervised methods by increasing dataset variety.

D. Contraction

Geometric verification checks the validity of the feature neighborhoods \( \hat{P}_{q_i} \) (or positive candidates from Section III-C) before merging elements in \( \hat{P}_{q_i} \) into \( P_{q_i} \). Equation (5) describes how the verified positives \( \hat{P}_{q_i} \) merges \( P_{q_i} \) during each epoch’s contraction step, where \( e \) represents the epoch number. The false positive in \( \hat{P}_{q_i} \) is noisy and might harm the learning process of the network.

Depending on different input types, the verification step varies and isolates from the network. In this letter, we use RANSAC for image verification and ICP for point cloud verification. The way of contracting positives is similar to the expansion of positive candidates. Similar to (3) and (4), we calculate the minimum matching score threshold \( \epsilon_i \) using RANSAC or ICP between the query and its temporal neighbors. Subsequently, we assess the matching scores between all positive candidates \( \hat{P}_{q_i} \) and the query. Any positive candidate with a matching score surpassing threshold \( \epsilon_i \) is classified as verified. Thus, verified positives \( \hat{P}_{q_i}^{(c)} \) are trustworthy and permanently added into the positive set \( P_{q_i} \):

\[
P_{q_i}^{(e)} = P_{q_i}^{(e-1)} \cup \hat{P}_{q_i}^{(c)}. \tag{5}
\]

E. Loss Function

We use triplet loss \( L_\theta \) for the \( i \)th training tuple \((q_i, \hat{p}_q^i, n_q^i)\) in our implementation as shown in (6), where \( \hat{p}_q^i \in \hat{P}_{q_i} \) and \( n_q^i \in N_q \). And \( P_{q_i} \) and \( N_q \) represent the positive and negative set updated from the previous iteration, \( m \) is hyperparameter (set to 0.2 by default). This loss is used by [7] to pull positive features together and repel negative features so that it will be easier to apply KNN and obtain the potential geographical nearest neighbors.

\[
L_\theta = \sum_{q_i} \left( \min_l \{ d_{f_o}^2 (q_i, \hat{p}_q^i) + m - d_{f_o}^2 (q_i, n_q^i) \} \right). \tag{6}
\]

IV. EXPERIMENTS

TF-VPR is tested in two kinds of environments: simulated RGB images [12], and real-world point cloud [13]. Additional real-world RGB experiments are provided in the appendix due to space constraints. Our codebase uses PyTorch [48] with network parameters optimized using Adam [49]. The learning rate is set to 1e-4 together with a weight decay of 1.0e-7. We compare TF-VPR with both supervised and self-supervised baseline methods.

Notation: We conducted a complete ablation study on both datasets. Any method incorporating any of the following modules should be regarded as TF-VPR. In the experiment section, we specifically denote TF-VPR as SPTM+A+F for convenience in the ablation study. Each module discussed in Section III is abbreviated as follows:

1) “+A” denotes the method with data augmentation.
2) “+F” denotes the method iteratively expanding spatial neighborhoods (including expansion and contraction).

A. Evaluation Metrics

Recall rate (Recall@N) measures the ratio of successful retrievals to the total number of queries. A retrieval is considered successful if at least one of the top-N retrieved results is a ground-truth spatial neighbor of the query. Ground truth can be obtained by K-D tree search on geographical location \((x, y, z)\) within a certain radius \( R \). It is important to exclude temporal neighbors of a query from its top-N retrievals when computing this metric. In our setup, temporal-based methods can easily overfit the temporal neighborhood, leading to uninformative evaluations with recall rates always at 100% if temporal neighbors are kept in the ground truth.

Heading diversity (HD) measures the diversity of sensor headings of the true positives (w.r.t. the query’s heading) among the top-\(|GT|\) retrievals. \(|GT|\) is the size of the ground-truth set based on a specific radius \( R \) as described in Recall@N. Those hard positives with headings different from the query are more valuable for loop closure in downstream applications [47]. Thus, to evaluate this heading diversity, we evenly divide the 360° range of headings into 8 angular bins. The first and last bins are excluded because they contain positives with similar headings w.r.t. the query. In this case, the \( m \)-th bin covers the heading...
difference range $Q_m$ as:

$$Q_m = [m \times 45^\circ, (m + 1) \times 45^\circ], m \in [1, 2, 3, 4, 5, 6].$$ \hspace{1cm} (7)$$

Then, we define HD for a query $q$ as the bin coverage ratio between the true positives and the ground truth:

$$\text{HD}(q) = \frac{1}{\varepsilon + \sum_{m=1}^{6} I(\exists x \in \mathcal{P}^{GT}_q \land (\theta_q - \theta_x) \in Q_m)},$$

$$\text{where } \theta_q \text{ and } \theta_m \text{ are respectively the heading of the query and a frame, } \mathcal{P}^{GT}_q \text{ is the set of true positives in the top-[GT] retrievals, } \mathcal{P}^{GT}_q \text{ is the ground truth positive set, } \varepsilon \text{ is an arbitrarily small positive quantity to avoid zero division error, and } I(\cdot) \text{ is the indicator function. See Fig. 4 for an example. Finally, we report the average HD for all queries.}

B. Experiments on KITTI-360 dataset

Dataset: KITTI-360 [13] dataset is an outdoor point cloud dataset captured with a Velodyne HDL-64E LiDAR and a SICK LMS 200 laser scanner in a roof-mounted pushbroom setup. This dataset comprises 100,000 laser scans covering a driving distance of 73.7 kilometers. We select three scenes for our study: Drive_0000 and Drive_0005 for training, and Drive_0002 for testing, containing 33,110 LiDAR scans. After pre-processing, we downsample each frame to 10,000 points, forming point clouds of size (10000, 4). Additionally, the GPS for each frame is provided for validation.

Baseline methods: The following baselines are evaluated:

1. PointNetVLAD [25] trained with pose-based supervision,
2. PointNetVLAD trained with temporal pseudo labels (SPTM) [10],
3. prototypical contrastive learning (PCL) [50] as another self-supervised VPR method used in visual navigation [29], and
4. the ablation study baselines.

Implementation details: Our implementation is based on PointNetVLAD. For training, we select $\delta = 5$ and $k = 2$ in (1) and (2), and sample 8 pairs of positives and negatives from the candidate pool for each query. For evaluation, we exclude the closest 30 temporal neighbors for each query from the top-N retrievals as explained in Section IV-A.

One baseline PCL is implemented with its default settings, tuning the total number of clusters. We set the number of clusters to 200, 500, and 1000, and select the best result.

Auto-labeling on KITTI-360: From Table I, SPTM+F shows superior performance in auto-labeling, outperforming SPTM with a 2% increase in recall rate and an 8% improvement in heading diversity. This improvement is due to the infrequent revisits of scenes from different angles in typical outdoor settings. Consequently, the addition of augmentation imposes an extra burden on the network, resulting in a decrease in both recall and heading diversity (HD).

Generalization on KITTI-360: SPTM+A and TF-VPR (SPTM+A+F) demonstrate strong generalization capabilities on the KITTI-360 dataset, as shown in Fig. 5. This improvement is due to the increased diversity in the training dataset through augmentation. It also quantitatively demonstrates a similar improvement in both recall rate and heading diversity for SPTM+A and TF-VPR. Furthermore, Table II demonstrates that TF-VPR outperforms other baselines in both scenes, achieving approximately 4% and 12% higher scores in terms of recall rate and heading diversity, respectively.

Compare TF-VPR with supervised methods: From Table II, we also observe that SPTM+A and TF-VPR comparable or even superior performance compared to the supervised method trained on the same dataset (small domain gap), significantly outperforming the one trained on the Oxford robot car dataset (large domain gap) [51]. This performance gap is due to the substantial domain differences between the training and testing datasets. Note that TF-VPR is a self-supervised method that allows us to collect data from similar scenes for self-supervised training. In contrast, other supervised baselines are often constrained by the availability of ground truth data (data privacy, sensor availability, annotation costs, and etc.). Therefore, when the domain gap between training and testing sets is small, TF-VPR data training holds an inherent advantage.
C. Experiments on Habitat-Sim Dataset

Dataset: TF-VPR is also tested via Habitat-Sim [12] simulator on the Gibson photorealistic RGB dataset [55], which offers panoramic RGB images for a variety of indoor scans. We capture RGB images with a panoramic camera mounted on a robot moving randomly in the virtual environment, resulting in a total of 33,679 RGB images across three Gibson rooms. Each image is downsampled to 256 × 64 pixels. In contrast to other datasets, this simulated RGB dataset contains a large number of revisits of places from both similar and different directions, which is useful for testing recall rate and heading diversity for VPR.

Baseline methods: Following Section IV-B, we use SPTM [10], NetVLAD [7], VLAD [4], [53], and PCL [50] as baselines. Additionally, we introduce the CNN image retrieval baseline CNNLR. [52] and a classic non-deep-learning VPR method VLAD [53] [4] for comparison.

Implementation details: We set δ = 5 and k = 2 in (1) and (2). For each query, we exclude its closest 30 temporal neighbors from the top-N retrievals as explained in IV-A. For the VLAD baseline, we use 128-dimensional SIFT features with a cluster size of 128. The raw VLAD descriptor dimension of 128 × 128 is reduced to 512 by PCA. Besides, CNNLR. uses pretrained model to extract the feature and follows the previous implementation for evaluation.

PCL, VLAD, CNNLR., and conventional visual SLAM system: Table IV shows the poor performance of PCL. It may not be ideal for VPR solutions because most contrastive learning methods tend to form disjoint clusters for each category, which may not effectively represent continuous features in vision-based SLAM problems. Contrarily, VLAD performs effectively as we aggregate features and evaluate performance on the same dataset, akin to network overfitting. However, the evaluation process is time-consuming compared to our learnable model, which only requires inference to a new model. Despite this, our method outperforms VLAD by approximately 8% in terms of recall rate. Furthermore, CNNLR., an image retrieval method, lacks the precision of most VPR methods and exhibits slightly inferior performance compared to SPTM. It also lags noticeably behind our proposed approach, particularly in terms of recall@1, where it trails by approximately 4%. Finally, we tested the conventional visual SLAM system, like OpenVSLAM [56]. However, OpenVSLAM easily loses track, with a total of 17.75% of frames lost during the tracking. OpenVSLAM constructs a disjoint topology graph while tracking odometry, resulting in a recall rate of 54.98% compared to 93.93% for TF-VPR. Due to its poor performance, we did not consider OpenVSLAM as a benchmark.

Auto-labeling on Habitat-Sim: Table III shows that TF-VPR, SPTM+A, and SPTM+F achieve the highest auto-labeling recall rate, with TF-VPR outperforming SPTM by about 1%. Similar to Section IV-B, feature neighborhoods(F) plays a critical role play in overfitting the test dataset. However, unlike the discussion in Section IV-B, augmentation enhances recall due to the abundant viewpoint variation and lack of lighting change in the simulated RGB dataset. Besides, TF-VPR outperforms all self-supervised baselines in heading diversity, with an improvement of around 8%.

Generalization on Habitat-Sim: In Table IV, TF-VPR outperforms all self-supervised baselines and approaches the supervised NetVLAD performance and VLAD+SIFT in both recall rate and heading diversity (HD). Besides, TF-VPR improves recall rate by 3% and heading diversity by 5%.

Compare TF-VPR with supervised methods: Similar to Section IV-B, Table IV indicates that the domain gap challenges the supervised method. SPTM+A and TF-VPR are comparable to the supervised method trained on the same dataset (small domain gap), outperforming the one trained on the Pittsburgh dataset (large domain gap) [51]. Additional per-frame performance visualizations are in the appendix.

Adaptability of TF-VPR on other backbones: We tested TF-VPR’s adaptability with backbones beyond NetVLAD and PointNetVLAD, showing significant improvements in heading diversity and recall rate. Table V shows that TF-VPR effectively applies to various backbones (MixVPR) [57] and PADLoC [58]) and modalities (point cloud and RGB), with each version outperforming its counterparts. This versatility enhances results across diverse scenarios.
The bottlenecks of TF-VPR models lie in their proc. int. conf. robot. automat. int. j. comput. vis. proc. conf. robot. learn. Proc. ieee/cvf conf. comput. vis. pattern recognit. conf. robot learn., 2022, pp. 429–443.

O. Vysotska and C. Stachniss, “Lazy data association for image sequences matching under substantial appearance changes,” IEEE Robot. Automat. Lett., vol. 1, no. 1, pp. 213–220, Jan. 2016.

N. Savinov, A. Dosovitskiy, and V. Koltun, “Semi-parametric topological memory for navigation,” in Proc. Int. Conf. Learn. Representations, 2018, pp. 1–10.

S. Garg, M. Vankadari, and M. Milford, “SeqmatchNet: Contrastive learning with sequence matching for place recognition & relocalization,” in Proc. Conf. Robot Learn., 2022, pp. 429–443.

V. C. Table V

| Data Type    | Metric | HD | R@1 | R@5 | R@10 |
|--------------|--------|----|-----|-----|------|
| RGB          | MixVPR | 63.47 | 48.01 | 79.66 | 86.50 |
|              | TF-VPR* | 70.98 | 53.91 | 82.74 | 89.46 |
| Pointcloud   | PADLoC | 25.42 | 39.47 | 44.71 | 46.34 |
|              | TF-VPR* | 35.42 | 43.73 | 49.22 | 51.19 |

* M represents TF-VPR model with MixVPR backbone and P represents TF-VPR model with PADLoC backbone. TF-VPR model is trained for 5 epochs.

We evaluate the auto-labeling capability of TF-VPR against two other backbones: MixVPR [57] and PADLoC [58]. The implementation details adhere to the original codebase and we use the pretrained models provided by the original authors for the baselines. Other abbreviations follow Table I.

V. CONCLUSION

We propose TF-VPR as a self-supervised VPR method adaptive to both auto-labeling and generalization tasks for determining the unknown spatial neighbors from the fixed temporal neighbors and learnable feature neighbors. While our method primarily utilizes panoramic RGB images and 360-degree point clouds, this iterative nature of TF-VPR allows for easy extension to non-panoramic inputs. Extensive experiments show that TF-VPR not only improves the recall rate over the existing method but also can retrieve spatial positives with more diverse viewpoints on various datasets. TF-VPR enables easier use of VPR in real-world robotics and computer vision applications, and can be applied to most existing deep-learning-based VPR methods.

Limitations: The bottlenecks of TF-VPR models lie in their generalization across large domain gaps. Specifically, one prevalent challenge in VPR is the impact of appearance changes, arising from factors such as lighting variations, weather, or seasonal transitions. TF-VPR shows limitations in generalization in these contexts and is more suitable for autolabeling tasks and VPR within the same scene.

ACKNOWLEDGMENT

The authors would like to thank Xuchu Xu for insightful discussions.

REFERENCES

[1] M. Zaffar et al., “VPR-Bench: An open-source visual place recognition evaluation framework with quantifiable viewpoint and appearance change,” Int. J. Comput. Vis., vol. 129, pp. 2136–2174, 2021.

[2] D. Sheng et al., “NYU-VPR: Long-term visual place recognition benchmark with view direction and data anonymization influences,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2021, pp. 9773–9779.

[3] D. G. Lowe, “Object recognition from local scale-invariant features,” in Proc. Int. Conf. Comput. Vis., 1999, pp. 1150–1157.

[4] R. Arandjelovic and A. Zisserman, “All about VLAD,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2013, pp. 1578–1585.

[5] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, “From coarse to fine: Robust hierarchical localization at large scale,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 12708–12717.

[6] Z. Chen et al., “Deep learning features at scale for visual place recognition,” in Proc. Int. Conf. Robot. Automat., 2017, pp. 3223–3230.

[7] R. Arandjelovic, P. Gronat, A. Torii, T. Papadakis, and J. Sivic, “NetVLAD: CNN architecture for weakly supervised place recognition,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2016, pp. 5297–5307.

[8] S. Garg, M. Vankadari, and M. Milford, “SeqmatchNet: Contrastive learning with sequence matching for place recognition & relocalization,” in Proc. Conf. Robot Learn., 2022, pp. 429–443.

[9] O. Vysotska and C. Stachniss, “Lazy data association for image sequences matching under substantial appearance changes,” IEEE Robot. Automat. Lett., vol. 1, no. 1, pp. 213–220, Jan. 2016.

[10] N. Savinov, A. Dosovitskiy, and V. Koltun, “Semi-parametric topological memory for navigation,” in Proc. Int. Conf. Learn. Representations, 2018, pp. 1–10.

[11] S. E. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, “SuperGlue: Learning feature matching with graph neural networks,” in Proc. Int. Conf. Robot. Automat., 2019, pp. 3223–3230.

[12] R. Arandjelovic, P. Gronat, A. Torii, T. Papadakis, and J. Sivic, “NetVLAD: CNN architecture for weakly supervised place recognition,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2016, pp. 5297–5307.

[13] S. Garg, M. Vankadari, and M. Milford, “SeqmatchNet: Contrastive learning with sequence matching for place recognition & relocalization,” in Proc. Conf. Robot Learn., 2022, pp. 429–443.

[14] O. Vysotska and C. Stachniss, “Lazy data association for image sequences matching under substantial appearance changes,” IEEE Robot. Automat. Lett., vol. 1, no. 1, pp. 213–220, Jan. 2016.

[15] N. Savinov, A. Dosovitskiy, and V. Koltun, “Semi-parametric topological memory for navigation,” in Proc. Int. Conf. Learn. Representations, 2018, pp. 1–10.

[16] S. E. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, “SuperGlue: Learning feature matching with graph neural networks,” in Proc. Int. Conf. Robot. Automat., 2019, pp. 3223–3230.
[20] T. Naseer, G. L. Oliveira, T. Brox, and W. Burgard, “Semantics-aware visual localization under challenging perceptual conditions,” in Proc. Int. Conf. Robot. Automat., 2017, pp. 2614–2620.

[21] S. Garg, N. Sudderhauf, and M. Milford, “Lost? appearance-invariant place recognition for opposite viewpoints using visual semantics,” in Proc. Robot. Sci. Syst. XIV, 2018, Art. no. 201.

[22] C. Choy, J. Park, and V. Koltun, “Fully convolutional geometric features,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2018, pp. 195–205.

[23] A. Forechhi, A. F. De Souza, C. Badue, and T. Oliveira-Santos, “Segmentation-appearanced-based global localization using an ensemble of KNN-DTW classifiers,” in Proc. Int. Joint Conf. Neural Netw., 2016, pp. 2782–2789.

[24] M. A. Uyy and G. H. Lee, “PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2018, pp. 4470–4479.

[25] K. Cai, B. Wang, and C. X. Lu, “Autoplace: Robust place recognition with low-cost single-chip automotive radar,” in Proc. Int. Conf. Robot. Automat., 2022, pp. 2222–2228.

[26] S. Lowry and M. J. Milford, “Supervised and unsupervised linear learning techniques for visual place recognition in changing environments,” IEEE Trans. Robot., vol. 32, no. 3, pp. 600–613, Jun. 2016.

[27] N. Maillet and G. Huang, “Lightweight unsupervised deep loop closure,” in Proc. Robot. Sci. Syst., 2018, pp. 1–8.

[28] O. Kwon, N. Kim, Y. Choi, H. Yoo, J. Park, and S. Oh, “Visual graph memory with unsupervised representation for visual navigation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 15870–15879.

[29] P.-Y. Lajoie and G. Beltrame, “Self-supervised domain calibration and uncertainty estimation for place recognition,” IEEE Robot. Automat. Lett., vol. 8, no. 2, pp. 792–799, Feb. 2023.

[30] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, “Learning with noisy labels,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2013, pp. 8135–8143.

[31] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, “Learning from noisy labels with deep neural networks: A survey,” IEEE Trans. Neural Netw. Learn. Syst., vol. 34, no. 11, pp. 8135–8153, Nov. 2023.

[32] Z. Yu et al., “Simultaneous edge alignment and learning,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 400–417.

[33] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, “Making deep neural networks robust to label noise: A loss correction approach,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2017, pp. 2233–2241.

[34] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, “Virtual adversarial training: A regularization method for supervised and semi-supervised learning,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 8, pp. 1979–1993, Aug. 2019.

[35] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, “Augmix: A simple data processing method to improve robustness and uncertainty,” in Proc. Int. Conf. Learn. Representations, Apr. 2020.

[36] H.-D. Lee et al., “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks,” in Proc. Int. Conf. Mach. Learn., 2013, Art. no. 896.

[37] Y. Zou, Z. Yu, B. Kumar, and J. Wang, “Unsupervised domain adaptation for semantic segmentation via class-balanced self-training,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 289–305.

[38] Y. Zou, Z. Yu, X. Liu, B. Kumar, and J. Wang, “Confidence regularized self-training,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 5982–5991.

[39] H. Yang, W. Dong, L. Carlone, and V. Koltun, “Self-supervised geometric perception,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 14350–14361.

[40] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 9726–9735.

[41] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” 2018, arXiv:1807.03748.

[42] Y. Tian, D. Krishnan, and P. Isola, “Contrastive multiview coding,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 776–794.

[43] S. Lowry et al., “Visual place recognition: A survey,” IEEE Trans. Robot., vol. 32, no. 1, pp. 1–19, Feb. 2016.

[44] A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla, “24/7 place recognition by view synthesis,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2015, pp. 1808–1817.

[45] C. Chen, X. Liu, Y. Li, L. Ding, and C. Feng, “DeepMapping2: Self-supervised large-scale LiDAR map optimization,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2023, pp. 9306–9316.

[46] A. Paszke et al., “Automatic differentiation in pytorch,” in Proc. NeurIPS Workshops, 2017.

[47] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Representations, 2015, pp. 1–10.

[48] J. Li, P. Zhou, C. Xiong, and S. C. Hoi, “Prototypical contrastive learning of unsupervised representations,” in Proc. Int. Conf. Learn. Representations, 2021, pp. 1–9.

[49] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, “1 year, 1000 km: The oxford robotcar dataset,” Int. J. Robot. Res., vol. 36, no. 1, pp. 3–15, 2017.

[50] F. Radenović, G. Toliša, and O. Chum, “Fine-tuning CNN image retrieval with no human annotation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 7, pp. 1655–1668, Jul. 2019.

[51] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, “Virtual adversarial training: A regularization method for supervised and semi-supervised learning,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 8, pp. 1979–1993, Aug. 2019.

[52] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, “Augmix: A simple data processing method to improve robustness and uncertainty,” in Proc. Int. Conf. Learn. Representations, Apr. 2020.

[53] D.-H. Lee et al., “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks,” in Proc. Int. Conf. Mach. Learn., 2013, Art. no. 896.

[54] Y. Zou, Z. Yu, B. Kumar, and J. Wang, “Unsupervised domain adaptation for semantic segmentation via class-balanced self-training,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 289–305.

[55] Y. Zou, Z. Yu, X. Liu, B. Kumar, and J. Wang, “Confidence regularized self-training,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 5982–5991.