PLSAV: Parallel loop searching and verifying for loop closure detection

Zhe Yang1 | Yun Pan1 | Lei Deng2 | Yuan Xie2 | Ruohong Huan3

1 College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China
2 Department of Electrical and Computer Engineering, University of California, Santa Barbara, California, USA
3 College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China

Correspondence
Yun Pan, postal: #1713, Yuquan Campus, Zhejiang University, No.38 Zheda Road, Xihu District, Hangzhou 310027, China.
Email: panyun@zju.edu.cn

Funding information
International Cooperation Research for Doctoral Students in Zhejiang University; Zhejiang Provincial Natural Science Foundation of China, Grant/Award Number: LY19F020032; Zhejiang Provincial Key Research and Development Program of China, Grant/Award Number: 2021C03027

Abstract

Visual simultaneous localization and mapping (vSLAM), one of the most important applications in autonomous vehicles and robots to estimate the position and pose using inexpensive visual sensors, suffers from error accumulation for long-term navigation without loop closure detection. Recently, deep neural networks (DNNs) are leveraged to achieve high accuracy for loop closure detection, however the execution time is much slower than those employing handcrafted visual features. In this paper, a parallel loop searching and verifying method for loop closure detection with both high accuracy and high speed, which combines two parallel tasks using handcrafted and DNN features, respectively, is proposed. A fast loop searching is proposed to link the bag-of-words features and histogram for higher accuracy, and it splits the images into multiple grids for high parallelism; meanwhile, a DNN feature extractor is utilized for further verification. A loop state control method based on a finite state machine to control these tasks is designed, wherein the loop closure detection is described as a context-related procedure. The framework is implemented on a real machine, and the top-2 best accuracy and fastest execution time of 80-543 frames per second (min: 1.84ms, and max: 12.45ms) are achieved on several public benchmarks compared with some existing algorithms.

1 | INTRODUCTION

The simultaneous localization and mapping (SLAM) systems have been developing rapidly in recent years, providing an estimation method of the pose and trajectory of moving agents including ground vehicles [1] and drones [2]. SLAM systems can make use of several kinds of sensors to build a graphical model of the environment, such as laser sensors [3], visual sensors [2], wireless sensors [4]. Among all the strategies of SLAM, the visual SLAM (vSLAM), which aims to structure the motion of the agents using only images captured by the camera, is one of the most popular approaches since it is inexpensive and common in realistic applications. Recent trends in multi-object tracking [5–7] and semantic mapping [8, 9] extend the capacity of sensing and understanding for SLAM systems, which significantly benefits some key emerging applications like safety monitoring and autonomous driving. Although the tracking and mapping tasks in SLAM can calibrate each other in a complementary mechanism, almost every vSLAM system suffers from accumulating errors coming from motion blur and illumination variance, and the most effective method to re-locate itself using images only is the loop closure detection [10], which is the main topic of this paper.

The loop closure detection, which can be found in many state-of-the-art vSLAM systems [2, 11] as an indispensable built-in function, is a task of recognizing previously visited places in a sequence of images and correcting the current position based on the previous location. The key algorithm for loop closure detection is place recognition including map-to-map (MtM), image-to-map (ItM), and image-to-image (ItI) methods. Williams et al. [12] compared different place recognition techniques and concluded that the ItI approach has better performance on accuracy for large-scale environments owing to its less intensive computation and memory usage, although sometimes it fails to achieve the best recognition accuracy because of lacking 3D map of the environment.

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2021 The Authors. IET Intelligent Transport Systems published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology
Therefore, this paper focuses on the ItI-based loop detection approach.

Existing ItI methods can be divided into two main categories according to the features extracted from the images: handcrafted and deep features. The handcrafted-feature-based methods extract raw features, or use the bag-of-words (BoW) \[10, 13–16\] method to convert raw features into brief vectors. On the other hand, recent advances in deep learning and convolutional neural networks (CNN) promote the development of deep-feature-based methods in loop closure detection, which however slows down the system due to the large consumption of computational and memory resources. This makes it difficult to implement a real-time system for loop closure detection with a deep feature extractor, especially on resource-limited platforms.

In our previous work \[17\], we presented gridding place recognition (GPR) for fast loop closure detection on mobile platforms, and the result showed that splitting the image with multiple grids can significantly speed up the loop closure detection procedure. In this paper, we aim to solve the ItI loop closure detection problem at an ultrafast speed, while the accuracy remains comparable. To this end, we propose a framework for loop closure detection, termed as parallel loop searching and verifying (PLSAV), combining the handcrafted and deep loop detection methods in a parallel structure with multithread computing, as shown in Figure 1. The framework of PLSAV mainly consists of two tasks: the task $S$ performing the fast loop searching (FLS) which reacts rapidly to the input frames with acceptable accuracy, and the task $V$ performing the deep verification (DV) using CNN features that periodically verifies the result. This structure is inspired by the parallel tracking and mapping (PTAM) \[18\] and parallel tracking and verifying (PTAV) \[19\] for SLAM and object tracking applications. In order to control the two parallel tasks, we propose a loop state control (LSC) method based on a finite state machine, describing the continuous loop closure procedure according to the temporal consistency. For a brief introduction of LSC, when a frame comes from an input sequence, the task $S$ first uses the FLS algorithm to find the best candidate that matches the current frame, that is, the place that is probably visited before. If the state in the LSC is "No Loop", the FLS algorithm will search for the candidate in all the previous frames, otherwise only a constant number of frames that are closed to last valid loop closure candidate will be checked, resulting in a significant acceleration to the detection. The result of the task $S$ will affect the state in the LSC, and also determine whether to send a request to the task $V$ to make a further confirmation about the loop closure using the DV algorithm. The final loop closure candidates will be corrected based on the states and the results of task $V$.

The main contributions of this paper are listed as follows:

1. The PLSAV framework is proposed for the ultrafast loop closure detection with comparable accuracy retained, using both the handcrafted loop detection and deep learning verification in a parallel structure. The LSC method is proposed based on a finite state machine to deal with these two parallel tasks, which describes the loop closure detection as a context-related procedure, regarding the temporal consistency as a strategy of detection rather than verification. We improve the CNN-based deep verification by centralization, making PLSAV framework as accurate as other state-of-the-art loop detection algorithms.

2. To speed up the loop detection, a loop closure detection method called FLS is proposed on the basis of a parallel similarity comparison of the handcrafted-based features and histogram of the input frames. For handcrafted-based features, the FLS splits the images into multiple grids to quickly extract the weighted and normalized BoW vectors in parallel. For the best-matched candidate retrieval, the LSC only evaluates several candidates simultaneously rather than searching among all the previous images in every frame. After these modifications the PLSAV can achieve less than 10 ms on processing a 640 $\times$ 480 image.

The performance of PLSAV on execution time and accuracy is shown by comparison with the existing loop closure detection approaches. In summary, the execution time of PLSAV achieves the fastest among all the methods, while the accuracy of PLSAV is comparable with the best reported ones.
The rest of this paper is organized as follows. In Section 2, an overview of existing loop closure detection methods is provided. The proposed PLSAV framework is presented in Section 3 with implementation details. Section 4 reports and discusses the experimental results on the accuracy and execution time. Finally, Section 5 concludes this paper.

## 2 RELATED WORK

ItI is the most efficient approach in loop closure detection. As mentioned in Section 1, ItI has two main categories: handcrafted-feature-based approaches and deep-feature-based approaches. Handcrafted features include BoW and raw features like oriented FAST and rotated BRIEF (ORB) [20]. Using raw features has better performance on the accuracy, however, the computational complexity increases dramatically when the size of the image dataset grows. Zhang et al. [21] proposed the bag of raw features to avoid the off-line vocabulary construction. They used direct feature matching to find the most similar image to the current input frame. Another substitute for direct feature matching is the locality sensitive hashing (LSH), which is used in the work by Shahbazi and Zhang [22]. However, the computation time and memory requirement for LSH-based methods still increase rapidly when the dataset expands. Apart from using points, Yin et al. [23] exploited a simple combination of endpoint and line feature to perform the loop detection. Although the processing speed is fast, the precision is lower than average level. In addition, patch- or grid-based methods [24–27] have gained more attention recently for performance over viewpoint variance. Garcia-Fidalgo et al. [24] used a hierarchical histogram of gridding images to generate a local feature like global descriptor, therefore the local descriptor and gridding global descriptor were combined as a concatenation rather than complementarity. In [25, 26], the authors did not consider the weight for different grids of the image, and the impact of the grid number was not discussed in the paper. Besides, the normalization of similarity for each grid is not defined, which will influence the threshold settings for different scenarios. Neubert et al. [27] split the images based on superpixel, which is less efficient for a fast front-end loop searching task, and the storage of irregular image patches becomes harder as the number of the images grows.

On the contrary, the BoW-based methods are more efficient than those using raw features. In the BoW method, the features extracted from a large dataset are clustered into different groups according to the similarity to form a vocabulary, then the test image is converted into a sparse numerical vector, representing the histogram of words present in the vocabulary. Although BoW has a good performance on computation time, it suffers from perceptual aliasing due to vector quantization [28]. Angeli et al. [13] proposed an online incremental BoW and used Bayesian filtering to estimate loop closure. Another two of the most acknowledged BoW-based methods are FAB-MAP [14] and DBoW2 [10, 15]. The FABMAP used the speed up robust features (SURF) and introduced the co-occurrence of words information into the observation likelihood using a Chow–Liu tree. The DBoW2 used ORB features to build a hierarchical vocabulary tree that discretizes a binary descriptor space under a fast speed. In order to refine the result, DBoW2 employed the temporal and geometrical consistency to check the validity of the loop closing pairs. A huge improvement on the accuracy can be seen when the spatial co-occurrence information is incorporated directly into the vocabulary [16]. Chen et al. [29] integrated the spatial information of every visual feature into BoW model to form an ordered word coding vector. The result showed that their method was faster and more accurate than the original BoW method. A more recent work [30] improved the incremental BoW method by deleting useless words rather than only adding the new words, and the traditional Bayesian filtering was changed to dynamic island method to reduce the processing time. The only drawback of the two new works is that the execution time increases significantly, which is over 200 ms on every image.

On the other hand, an increasing number of works focus on using deep learning framework to extract robust deep features rather than handcrafted features from input images over the last few years. In the general case of deep-feature-based loop closure detection, a neural network trained for object detection or image classification is applied to the whole image, and the output of a specific layer is used as a description vector for similarity comparison. Chen et al. [31] is claimed to be the first work to exploit the deep features extracted by CNN with a spatial and sequential filter to perform place recognition. The approach failed to achieve real-time performance, with the ability to process 2.5 frames per second (fps). Sunderhauf et al. [32] used CNN to identify landmarks in the images rather than feature points, addressing the viewpoint and condition invariance challenges. However, the execution time is 1800 ms per image on a GPU-accelerated desktop, which is not fast enough to perform a real-time detection. Besides, large viewpoint change sometimes indicates a large displacement, which leads to bad performance in some strict localization applications. Hou et al. [33] exploited a pre-trained CNN model for image classification to perform the loop closure detection task. They demonstrated the performance using features from different layers in CNN and provided a detailed evaluation. The execution time for feature extraction only is 155 ms per frame on CPU and 19 ms per frame on GPU. Apart from the CNN model, the generative model is another feasible loop closure detector using deep learning. Zhang et al. [34] took the discriminator out of a generative adversarial network (GAN) as a loop closure detector, and the limited result indicated that the GAN-based method outperforms the BoW method. Kim et al. [35] clustered images to a similar size of superpixels and detected the boundaries as feature points. Then, a DCGAN model was used to perform as a discriminator in the recognition process, which was claimed to outperform the BoW method as well. Shin et al. [36] also used the local patch descriptor from GAN to perform loop closure detection and provided comprehensive results, however the execution time for each frame was 1956.9 ms, which is 30x slower than the CNN-based method [33], and the average precision is just comparable.
Meanwhile, a variant of ItI method called sequence-to-sequence (S2S) loop detection method [37–40] is also popular owing to its efficiency to deal with multiple images at the same time. Arroyo et al. [37] summarized a sequence of images into binary vectors, and the best loop candidate could be found to have the smallest hamming distance. Since the S2S method relies heavily on the trajectory-based loop rather than point-based loop, the performance is bad in some scenarios with unstable image sampling rate, thus Bampis et al. [38] extended the S2S to a combination of bag of sequence words and bag of image words, which takes advantages from ItI and S2S methods to form a robust two-layer loop detection. Li et al. [39] made a combination of holistic feature and local features to perform S2S loop matching, but the result was not as good as [38]. However, S2S method is not good at dealing with sharp turns or the overlap between sequences. These issues originated from the fact that S2S loop detection is based on a fixed length of the sequence. Thus, ItI is still the most common and practical choice for loop closure detection in state-of-the-art SLAM systems [2, 11], and most of the standalone loop closure detectors [10, 13–16, 30] adopted ItI for place recognition, which makes it portable and extensible for different kinds of vSLAM systems for further integration.

Recently, several works made attempts on the acceleration of loop closure detection to provide execution time efficiency using the extra hardware. Kim et al. [41] utilized general-purpose computing on graphics processing units (GPGPU) technique to average the descriptors of feature points at each frame. The experimental results demonstrated that this approach boosted the speed compared with conventional methods. Bampis et al. [42] proposed a local color and texture descriptor (LoCATe) to quickly summarize the image using GPGPU. The extracted features were then sent to the DBoW2 framework to perform the loop closure detection. Although the execution time was sped up by utilizing the extra hardware, these methods had poor results on the accuracy, which were below average performance.

Our proposed method, PLSA V, is constructed using both handcrafted-feature-based and deep-feature-based approaches. In PLSA V, the front-end task FLS is an extension of BoW-based loop closure detector, where the DBoW2 library [15] is used to train and convert the image into a binary BoW representation. Compared with other gridding-based methods [24–27], we increase the accuracy and speed of the loop detector by extracting the weighted and normalized BoW vectors in parallel and combining these local descriptors with a global histogram descriptor. The back-end task in PLSA V is a CNN-based place recognition algorithm inspired by the structure in [33], with the extension for the definition of similarity check and the method of score normalization. Different from other parallel frameworks for machine vision task like [18, 19], PLSA V has a bidirectional communication between the front-end task and back-end task, where the feedback from the verification task can change the behavior of the loop searching task to improve the speed and accuracy of the loop detection, and it will affect the trigger of verification in turn. More details will be discussed in the following sections.

3 METHODS

3.1 Fast loop searching

The overview of FLS is shown in the Figure 2, which is derived from our previous work GPR. In the GPR, inspired by Lazebnik [43], we split the images into multiple grids for high parallelism and use BoW features to summarize each grid. The matching results for each grid are then retrieved and merged for further determination of loop closure under extra temporal and spatial constraints. To form the BoW using the pre-trained vocabulary, we adopted ORB feature because of its ability to cope with changes in rotation and scale, together with the fast speed of feature detecting and matching. Assume that the image $I_i$ is partitioned into $N$ grids at the time step $i$, and each image patch $I_{i,n}$ has a BoW vector $v_{i,n}$ at the grid $n$, with the number of ORB features equaling $f_{i,n}$. The L1-score similarity $s_{BoW}$ between two images $I_i$ and $I_j$ can be described in (1):

$$
v_j = \frac{\sum_{k=1}^{N} f_{i,k} v_{i,k} v_{j,k}}{\sum_{k=1}^{N} f_{i,k}}
$$

$$
s_{BoW}(I_i, I_j) = 1 - \frac{1}{2} \left( \frac{|v_i - v_j|}{|v_i| |v_j|} \right).
$$

Unlike the GPR, the FLS uses $\sum_{k=1}^{N} f_{i,k}$ to balance the weight between the grids according to feature numbers. To normalize the range of scores with the best score expected in the current image sequence, the normalized similarity score $\eta_{BoW}$ is governed by (2), where $\mu_{BoW}$ is the normalization factor as described in (3) when $i > 3$:

$$
\eta_{BoW}(I_i, I_j) = \frac{s_{BoW}(I_i, I_j)}{\mu_{BoW}(I_i)}.
$$

$$
\mu_{BoW}(I_i) = \max \left( s_{BoW}(I_i, I_{i-1}), \sum_{k=1}^{i-1} s_{BoW}(I_k, I_{k-1}) \right). \tag{3}
$$

where $s_{BoW}(I_i, I_{i-1})$ is the one used in original DBoW2 and the second item is set here to avoid $\mu_{BoW}(i)$ being erroneously small, which may potentially occur in huge turning scenarios. In the original DBoW2, the algorithm will discard $\eta_{BoW}$ when the $\mu_{BoW}(i)$ is too small, however in the proposed PLSA V $\mu_{BoW}(i)$ is defined as the similarity between the current and previous frame or the average of the valid similarity between successive image pairs seen in the previous frames, which is updated whenever a new frame is captured and tends to be stable as the sequence gets larger. Thus, the normalized similarity is available in every frame, avoiding the discarding which happens frequently in DBoW2. Based on the normalized score, we can
obtain the best candidate for loop closure among all the previous frames. Although the distribution of feature points becomes even when the entire image is partitioned into several grids, the number of total feature points may decrease since we set the oriented FAST (oFAST) detection threshold in each grid the same in order to retain the discrimination power of each feature point. Therefore when an image of small size is divided into pieces, the description for each grid becomes vague without enough texture information, while in datasets with large image size the gridding scheme is acceptable.

To quickly evaluate the validity of loop closure pairs between two images $I_i$ and $I_j$, the task $S$ first converts the original image to an illumination invariant image, then the 255-bin greyscale histogram of the converted image is calculated and flattened into vectors. According to [37], the illumination invariant image is calculated as follows:

$$I = \log(G) - \gamma \cdot \log(B) - (1 - \gamma) \cdot \log(R),$$

$$\lambda_G \frac{\gamma}{\lambda_B} + \frac{(1 - \gamma)}{\lambda_R},$$

where $R, G, B$ are the color channels of the input frame and $I$ is the resultant illumination invariant image. $\lambda_G, \lambda_B, \lambda_R$ are peak spectral responses of each color channel, which can be obtained in camera specifications.

The histogram vectors will be further compared using the Pearson correlation coefficient $\epsilon_{hist}(I_i, I_j)$ with the normalization factor $\mu_{hist}$, as shown in (5)–(7) when $i > 3$, where the $\mu_{hist}$ is similar to $\mu_{BoW}$. $\bar{h}_i$ is the mean of the histogram vector $h_i$ and “." is the dot product operation.

$$\epsilon_{hist}(I_i, I_j) = \frac{(h_i - \bar{h}_i) \cdot (h_j - \bar{h}_j)}{\|h_i - \bar{h}_i\|_2 \|h_j - \bar{h}_j\|_2},$$

$$\mu_{hist}(I_i) = \min \left( \epsilon_{hist}(I_i, I_{i-1}), \sum_{k=2}^{i-1} \epsilon_{hist}(I_{k}, I_{k-1}) \right).$$

The calculation of histogram and the multiple subtasks for ORB feature extraction in different grids can be executed in parallel on a multicore system to speed up the FLS algorithm. Note that the histogram only carries global texture information without any shape or structure information, so it is sensitive to illumination variance. The illumination invariant conversion addresses the problem, making the histogram a great complementary knowledge for BoW vectors.

As can be seen from Figure 2, when the state in PLSA $V$ remains at $S_1$ or $S_2$ rather than jumping to $S_0$, FLS will not ‘check start’ by searching from all the previous frames, but only some candidates around last loop candidate, coined as fast candidate” in this paper. More details about the two functions of loop searching and the state control mechanism can be found in Section 3.3.

### 3.2 Deep verification

We use Tensorflow [44], an open-source software library for machine learning applications such as neural networks, to extract deep features in the DV algorithm implementation. The Places [45] dataset with over 2.5 million images of 205 scene categories is used to train the network in this paper. In order to achieve both speed and accuracy [45, 46], we exploit a
publically available VGG16 [47] model, which has been pre-trained on the Places dataset, to extract deep features for loop closure verification. There are 13 convolutional layers, 5 max-pooling layers, and 3 fully connected layers, as shown in Figure 3 and Table 1. We name the max-pooling layers as POOL1, POOL2, ..., POOL5, and the fully connected layers as FC6, FC7, FC8. We take the results from the last convolutional layer in every block and name them as CONV1, CONV2, ..., CONV5. Once the feature map is extracted from a certain layer, it is flattened to a vector and normalized using L2-norm, then used as the high-level feature description of the image.

Note that max-pooling layers always reduce the dimensions of the deep features from convolutional layers by using down-sampling to merge local features. This procedure builds a much more concise representation of the image and provides translation invariance. However, the fully connected layers, also with lower dimensions of the deep features, lose spatial information for the purpose of high-level reasoning. Though this is good for object detection and image classification, it will cause inaccurancy in loop closure detection tasks since the geometrical information is indispensable.

Whenever task $\mathcal{Y}$ receives a request from the task $\mathcal{S}$ to verify the validity of the loop candidates image $I_i$ and image $I_j$, it always performs the DV algorithm and ignores the following requests from the task $\mathcal{S}$. In the DV algorithm, the deep features $d_i$ and $d_j$, from the two images $I_i$ and $I_j$, will be first extracted in parallel. Then, different from the traditional CNN-based method like [33], the deep similarity score $\eta_{\text{CNN}}$ between the two feature vectors will be calculated using Pearson correlation coefficient $\mu_{\text{CNN}}$ with the normalization factor $\mu_{\text{CNN}}$, as shown in Equations (8) and (9), where $\bar{d}_i$ denotes the mean value of the elements of $d_i$ and “$\cdot$” is the dot product.
operation. The Pearson correlation and the normalization factor can provide a centralized result that will eliminate the inherent discrepancy between different scenes.

\[ \eta_{\text{CNN}}(I_i, I_j) = \frac{\epsilon_{\text{CNN}}(I_i, I_j)}{\mu_{\text{CNN}}}, \]

\[ \epsilon_{\text{CNN}}(I_i, I_j) = \frac{(d_i - \bar{d}_i) \cdot (d_j - \bar{d}_j)}{\|d_i - \bar{d}_i\|_2 \|d_j - \bar{d}_j\|_2}. \]

The \( \mu_{\text{CNN}} \) is initialized at the beginning of the PLSAV as 1 and updated using the average of all the available deep feature correlation coefficient between the successive image pairs until the first request received from the task \( S \). We set a threshold on the coefficient, denoted as \( \theta_{\text{CNN}} \), to determine if the loop closure candidates are valid. Candidates with \( \eta_{\text{CNN}} \) greater than \( \theta_{\text{CNN}} \) will be accepted as valid ones, otherwise, they will be rejected.

### 3.3 Loop state control

Whenever a loop closure is detected, consequent images in the sequence are more likely to be regarded as loop closing points, which is a phenomenon referred to as the temporal consistency for loop closure detection. The temporal consistency can be either a constraint to determine when the loop closure actually occurs, or the assumption that the following temporally consistent image pairs are probably the loop closing pairs. The temporal consistency is constructed as a group matching method in [10, 15]. In our method, we use the LSC to describe the continuous loop closure detection as a context-related procedure, as illustrated in Figure 4. Figure 5 presents an example of the LSC procedure.

We define \( i \) as the current image index, \( i_{\text{last}} \) as the last image index when the loop closure happened, and \( i_{\text{last}} + 1 \) as the corresponding match index for \( i_{\text{last}} \). There are three states in the LSC. The first state “No Loop” (state S0) corresponds to the case that no valid loop closure candidate is found among all previous images under FLS for the current image. In detail, the criteria for the loop closure at this state are \( \eta_{\text{BoW}} > \theta_{\text{BoW}} \) and \( \eta_{\text{hist}} < \theta_{\text{hist}} \). Once a valid loop closure candidate with index \( l \) is found, the state transfers to “Initial Loop” (state S1). Meanwhile, the task \( S \) will send a request to the task \( V \) to make a further confirmation of the loop closure between the \( I_l \) and image \( I_{i_{\text{last}}} \). If the task \( V \) finds that the loop closing pairs were detected erroneously, all the loop closing pairs in the following sequence before completing the task \( V \) will be discarded as invalid loop closure detection.

We set a variable \( t \) to count the number of temporally consistent matches when the state is in “Initial Loop”. In this state, the task \( S \) only has to search for candidates in several images around the last matched image for loop closure detection, that is, \( i_{\text{last}} \). In this paper, we choose four images with indices \( i_{\text{last}} - 1, i_{\text{last}} + 1, i_{\text{last}} + 2 \), to find the one most similar with the current image. If the task \( S \) fails to find a valid loop closing candidate, the state will go back to “No Loop” and \( t \) is reset to zero. When the task \( S \) finds the candidate and \( t \) reaches \( \theta_t \), the minimum number for temporal consistency, the state will transfer to “Inside Loop” (state S2). This means that if several \( \theta_t \) adjacent images can find \( \theta_t \) temporally consistent loop closure candidates, the following images in the sequence will probably find loop closure candidates as well, continuously.

In the state S2, the task \( S \) only compares the similarity using four images, like the process in the state S1, but the criteria to find the valid candidate is more relaxed as \( \eta_{\text{BoW}} > \theta_{\text{BoW}} / \lambda \) and
\[ \eta_{\text{hist}} < \frac{\theta_{\text{hist}}}{\lambda}, \] where \( \lambda \) is the scale factor to lower the bar of finding the right loop closing pairs. When the task \( S \) finds the most similar image \( l \) of the current image \( i \), but fails to pass the criteria, a request will be sent to the task \( V \) to check whether the image \( i \) and image \( l \) are actually paired.

Even when the task \( S \) fails to detect a valid candidate, the state can still remain in “Inside Loop” until \( i - i_{\text{last}} \) is greater than \( \theta_{\text{Detection}} \), which means that too many images have no loop closure candidate since the last successful detection happened. \( \theta_{\text{Detection}} \) is the maximum number of images that fail to detect loop closure in the state S2. Whenever the gap between \( i \) and \( i_{\text{last}} \) is over \( \theta_{\text{Detection}} \), the task \( S \) re-detects the loop closure candidate under strict criteria among all the previous images, then the state will transfer back to “No Loop” if the detection fails, and “Initial Loop” if otherwise.

The entire PLSA V framework can be summarized to the pseudo-code presented in Algorithms 1 and 2, where \( N \) is the number of grids in the FLS, \( is\text{-}Lock \) state refers to the availability of task \( V \), and \( is\text{-}Loop \) is the result matrix for loop closure detection. Figure 6 demonstrates an example of state transfer for the algorithm running on datasets NewCollege (NC) [48] from image index 950 to 1050.

### 3.4 EXPERIMENTAL RESULTS

#### 3.5 Datasets

We use the eight most widely used datasets to comprehensively evaluate our algorithm, including NC [48], CityCentre (CC) [14], Malaga6L (M6L) [49], LipOutdoor (LO) [13], LipIndoor (LI) [13], and KITTI [50]. Table 2 presents the brief properties of the five datasets containing different kinds of outdoor and indoor scenarios, contributing to the performance of the real-life application. The NC and CC datasets originally provide the ground-truth reference for loop closure detections as a \( M \times M \) binary matrix, where \( M \) is the number of images in the dataset. If the value of the element at row-\( x \), column-\( y \) equals 1 in the matrix, the \( x \)th image and \( y \)th image in the dataset can be regarded as loop closure pairs, otherwise a zero value indicates that they were not captured at the same place. The ground-truth data for M6L, LO, and LI can be found in [51] as matrix files as well, while the ground-truth data for KITTI is derived from [52]. Note that the left and right images in NC and CC are processed separately, which is the same as the first experiment in [16] and different from the settings in [51]. For M6L we only choose the left camera sequence because images from the left and right cameras are too close to be considered as two different datasets. Since not all the sequences in KITTI have loop
ALGORITHM 2 Some detailed functions in Algorithm 1

Function checkStart(i):
1. Find the best match index j in $s_{bow}(I[i])$ for j in $[0, i - 1]$ using bow[i] and bow[j].
2. Calculate $\eta_{bow}(i, j)$ and $\eta_{hist}(i, j)$.
3. if $\eta_{bow}(i, j) > \theta_{bow}$ and $\eta_{hist}(i, j) < \theta_{hist}$ then dispatch thread: $taskV(i, j)$.
end
4. detectLoop(i, j)
5. state $\leftarrow$ 1
6. $t \leftarrow t + 1$
end

Function fastCandidate(i, j):
1. dispatch threads $p$ in $[0, 3]$: Calculate $\eta_{bow}(i, j - 1 + t)$.
2. Wait for synchronization with threads $p$ in $[0, 3]$
3. return arg max$_k \eta_{bow}(i, k)$ for $k$ in $[j - 1, j + 2]$

Function detectLoop(i, j):
1. isLoop[i][j] $\leftarrow$ true
2. $i_{start} \leftarrow i$
3. $l_{start} \leftarrow l$

Function taskV(i, j):
1. isLock $\leftarrow$ true
2. Calculate $\eta_{CNN}(i, j)$ using (8)(9)
3. if $\eta_{CNN}(i, j) < \theta_{CNN}$ then $i_{deep} \leftarrow i$
4. $l_{deep} \leftarrow l$
5. $isLock \leftarrow false$

TABLE 2 Details of the datasets for experiments

| Dataset | Camera | Description | Image size (pixels) | Number of images |
|---------|--------|-------------|---------------------|------------------|
| NC      | Lateral| Outdoors    | 640×480             | 2146             |
| CC      | Lateral| Outdoors    | 640×480             | 2474             |
| M6L     | Lateral| Outdoors    | 1024×768            | 3474             |
| LO      | Frontal| Outdoors    | 240×192             | 531              |
| LI      | Frontal| Indoors     | 240×192             | 388              |
| K00     | Frontal| Outdoors    | 1241×376            | 4541             |
| K05     | Frontal| Outdoors    | 1226×370            | 2761             |
| K06     | Frontal| Outdoors    | 1226×370            | 1101             |

We build part of the task $S$ in the proposed PLSA framework based on the DBoW2 library [15] with the function of training and converting the image into a binary BoW representation. The original settings of ORB features in [15] are set to extract 1000 key points per image, at 8 different scales with a scale factor of 1.2. While in the FLS algorithm the image is divided into $N$ grids, so we only extract $1000/N$ key points in each grid retaining the same oFAST threshold. We train the vocabulary using over 10,000 images from the dataset Bovisa closure, and the number of loop closure events is small in some sequences, we only choose KITTI 00 (K00), KITTI 05 (K05), and KITTI 06 (K06) in the manuscript as they contain the most ‘revisited’ locations [30, 53].

We implement our PLSA V framework using C++ with multi-threading on an AMD RYZEN 1950X @ 3.4 GHz machine. As a background task, the average execution time of task $V$ 114.35 ms using the Tensorflow framework without GPU support, and the number decreases to 18.66 ms when a single NVIDIA GTX 1080Ti GPU is added to speed up the task. To make a fair comparison, we use the result without GPU support. The performance of a loop closure detection system is evaluated by measuring the recall and precision of the loop closure detection. Precision means the fraction of true positive loop matches among all the loop closures detected by the algorithm, while recall is the ratio between true positive loop matches and all the loop closure events taken place in the ground truth of the datasets. The accuracy of loop closure detection is defined as the maximum recall rate under 100% precision since any false positive hit will significantly reduce the performance of the localization system, which is not allowed in real-world applications.

In order to determine the parameters in PLSA V, we initially test the framework on the BO dataset off-line for several times to obtain the best recall rate under 100% precision, and the best parameters were further used in PLSA V evaluation for accuracy and speed with other works. Since the accuracy definition of the loop closure detection is not like a typical two-class classification problem and the correlation between the parameters and accuracy is non-linear, we used an exhaustive grid searching over specified parameter values. In detail, we varied $\theta_{bow}$ from 0 to 0.5 with each step of 0.05, $\theta_{hist}$ from 0 to 1 with each step of 0.15, $\theta_{CNN}$ from 0 to 1 with each step of 0.15, $\theta_D$ from 1 to 5 with each step of 1, $\theta_{fake}$ from 1 to 5 with each step of 1, and $\lambda$ from 1 to 3 with each step of 0.5. The parameters are set as $\theta_{bow} = 0.5$, $\theta_{hist} = 0.35$, $\theta_{bow} = 0.35$, $\theta_{hist} = 0.75$, $\theta_{CNN} = 0.5$, $\lambda = 2$ in the following experiments.

We first evaluate the performance of the loop closure detection using features from different layers in the VGG16 architecture to determine the best layer for the DV task. We vary the correlation

![FIGURE 7](image-url)
threshold $\theta_{\text{CNN}}$. Note that the FC7 and FC8 layers are excluded since the result is too bad with extremely low precision, and FC6 never achieves 100% precision. For detailed evaluation, Table 3 shows the accuracy (recall rate under 100% precision) of loop closure detection for different layers on the CC dataset. According to the results, the performances of convolutional and max-pooling layers are increasing layer by layer, while the fully connected layers suffer from poor accuracy since the spatial information increases when the convolutional layer gets deeper in the neural network architecture, which does not happen in the fully connected layers where the high-level reasoning is the main purpose. Thus, taking the performance and the feature size into consideration, POOL5 is chosen for the feature layer of the DV in this paper owing to its best accuracy and fewer feature dimensions. The average precision of POOL5 for CC and NC datasets are 0.8930 and 0.9311 in DV, compared to 0.8480 and 0.8242 in [33]. The improvement is introduced by the Pearson correlation measurement and the normalization factor.

On the other hand, the gridding number in the FLS algorithm has a big impact on the performance of loop closure detection as well. Figure 8a,b shows the impact of gridding number on PR curves for NC and CC datasets in 4, 9 and 16 grids situation, varying $\theta_{\text{BoW}}$ from 0.25 to 0.35 with each step of 0.01. Note that the term “CC-N” indicates that the FLS algorithm divides the image into $N$ grids in the PLSAV framework on the CC dataset, while “NC-N” has the same definition on the NC dataset. Besides, Figure 8c,d presents the processing time of 200 frames (from 400 to 600) of the NC and CC datasets in 4, 9, and 16 grids situation. According to the result, the accuracy of loop closure detection decreases when the number of grids increases, which however boosts the execution speed. Thus, there is a trade-off between the accuracy and execution time with different gridding numbers in FLS. Since the 4-grid situation is fast enough to perform a real-time detection with the best accuracy, we choose the PLSAV system which has the FLS algorithm with the 4-grid scheme as our final solution, which will be used in the following part.

### 3.7 Accuracy

As mentioned above, the accuracy of the loop closure detection is the maximum recall rate under 100% precision. Table 4 presents the results of comparison among PLSAV and other algorithms including direct feature matching [22], incremental BoW [13], FAB-MAP with SURF [14], DBoW2 with FAST [10], DBoW2 with ORB [15], IBuILD [55], RTAB-MAP [51], BoW pairs [16], CNN-based approach [33], incremental binary BoW [30], and our gridding BoW method GPR [17]. Note that “-” in the table means no results provided on the given dataset. The result for [14] on the KITTI dataset is obtained from [24]. For each dataset, we highlight the top-2 accuracy among all the algorithms. It can be seen from Table 4 that PLSAV achieves the top-2 best recall rates in all datasets under 100% precision, and in NC, CC, and LO datasets the PLSAV has comparable results with the best accuracy.

---

### Table 3: Comparison of the accuracy using different layers in VGG16 architecture on the CC dataset

| Layers  | CONV1 | POOL1 | CONV2 | POOL2 | CONV3 | POOL3 | CONV4 | POOL4 | CONV5 | POOL5 | FC6, FC7, FC8 |
|---------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|---------------|
| Accuracy | 0.1373 | 0.1800 | 0.2246 | 0.3030 | 0.3102 | 0.3226 | 0.5187 | 0.5615 | 0.5579 | 0.5900 | -              |

---

Note that the term “CC-N” indicates that the FLS algorithm divides the image into $N$ grids in the PLSAV framework on the CC dataset, while “NC-N” has the same definition on the NC dataset.
TABLE 4
Comparison of the accuracy of different algorithms

|       | Shahbazi [22] | Angeli [13] | Cummins [14] | Mur-Artal [15] | Khan [55] | Labbe [51] | Kejriwal [16] | Hou [33] | Garcia-Fidalgo [30] | Yang [17] | PLSAV |
|-------|---------------|-------------|-------------|---------------|-----------|-----------|--------------|--------|---------------------|----------|-------|
| NC    | 48.8          | -           | 47          | 70.29         | -         | 59.78     | -            | 79.40  | 71.28               | 72.50    |
| CC    | 54.18         | -           | 38.77       | 43.03         | 38.92     | 66.8      | [45, 50)      | 88.25  | 50.36               | 87.70    |
| M6L   | -             | -           | 68.52       | 81.51         | 78.13     | -         | 87           | -      | 82.76               | 91.38    |
| LO    | 45.3          | 71          | -           | 25.5          | 70        | 90.9      | -            | 85.24  | 72.59               | 88.04    |
| LI    | 67.3          | 80          | 23.64       | 41.9          | 85        | 94        | -            | 83.18  | 75.11               | 94.47    |
| K00   | -             | -           | 49.21       | -             | -         | -         | -            | 76.50  | 75.09               | 84.86    |
| K05   | -             | -           | 32.15       | -             | -         | -         | -            | -      | 63.18               | 71.46    |
| K06   | -             | -           | 55.34       | -             | -         | -         | -            | 95.53  | 88.56               | 97.05    |

FIGURE 9
The PR curves of PLSAV, DBoW2, and FABMAP on (a) NC, (b) CC, (c) K00, and (d) K06 dataset, where “H” means histogram and “DV” means deep verification.

The improvement gained by the histogram and DV is shown in Figure 9, where the PR curves on the NC, CC, K00, and K06 datasets are plotted with different components missing in the PLSAV framework. The curves from DBoW2 and FABMAP are also included by running the open-source codes on the same machine, while the other methods in Table 4 are not publicly accessible and thus, they are not plotted in the figure. It is observed that the FLS loses accuracy for faster candidate retrieval, however the performance increases dramatically when the DV is included, indicating the great impact of deep features in the PLSAV framework. Nevertheless, the weighted and normalized gridding BoW approach with the loop state control has better performance than DBoW2 and FABMAP, as more loop pairs were found under 100% precision.

Part of the loop pair query matrices for different algorithms are drawn in Figure 10. The x-axis and y-axis in each subfigure denote the index number of the image in the dataset, note that the matrices are zoomed in to better illustrate the distinction. In each matrix, if the algorithm finds the xth image and yth image in the dataset are loop pairs, then the pixel in row-x, column-y will be white, otherwise black. At each row, the left, middle, and right columns are the results from DBoW2, CNN-based detector, and PLSAV, respectively. All the query results are obtained by achieving the best recall rate under 100% precision. Owing to the temporal consistency provided by the relaxed validation from the state S1 and S2 in LSC, and the improved similarity normalization in FLS and DV, the PLSAV can continuously recognize loop pairs in long trajectories regardless of pure rotations and viewpoint variance as shown in the green boxes at the right column, while these loop pairs are missed by DBoW2 and CNN-based detector in the red boxes at the left and middle columns. Besides, weighted gridding features in FLS can avoid the disturbance by sudden moving objects in the images, which increases the recall rate for PLSAV. Figure 11 shows some examples of loop pairs that were detected by PLSAV but missed by DBoW2 and CNN-based detector. Viewpoint variance occurs in Figure 11a,b, and the pure rotation that wrongly reduces the similarity in DBoW2 happens in Figure 11c. Figure 11a,b,d have
moving objects like pedestrians and cars that can easily distract the loop closure detector from its local and global perception.

### 3.8 | Execution time

We compare the execution time of PLSAV with the existing methods on the above five datasets, as shown in Table 5. Each cell of the table presents the average processing time on each frame of the sequence in milliseconds (ms). “-” in the table means no results are provided on the dataset. Number with “>” means only part of the algorithm has a measurement for execution time, for example, the feature extraction procedure. Note that Table 5 does not include the slight delay between the corrected result (from task $\mathcal{V}$) and the immediate result (from task $\mathcal{S}$), which can be omitted in localization systems where
TABLE 5 Comparison of the execution time of different algorithms in milliseconds

|       | Angeli [13] | Cummins [14] | Mur-Artal [15] | Khan [55] | Labbe [51] | Kejriwal [16] | Hou [33] | Garcia-Fidalgo [30] | Yang [17] | PLSAV |
|-------|-------------|--------------|----------------|----------|-----------|--------------|---------|---------------------|---------|------|
| NC    |             |              | 22             | -        | 603       | 393          | >19     | 352.08              | 5.07    | 8.05 |
| CC    | 5820        | 466          | -              | 1330     | 441       | 368.41       | 5.25    | 7.13                |
| M6L   | -           |              | 50             | -        | -         | -            | -       | -                   | 8.54    | 12.45 |
| LO    | -           |              | -              | -        | 404       | 120          | -       | 249.45              | 1.92    | 2.89 |
| LI    | 255         | 588          | -              | 141      | 69        | 19.17        | -       | 0.98                | 1.84    |
| K00   | -           |              | 237.4          | -        | -         | -            | -       | 432.38              | 6.12    | 8.34 |
| K05   | -           |              | 233.8          | -        | -         | -            | -       | -                   | -       | 6.03 |
| K06   | -           |              | 188.2          | -        | -         | -            | -       | 395.16              | 4.99    | 6.67 |

(a) Loop pairs in NC

(b) Loop pairs in CC

(c) Loop pairs in K00

(d) Loop pairs in K06

FIGURE 11 Some examples of loop pairs in (a) NC, (b) CC, (c) K00, and (d) K06 dataset from Figure 9 that were detected by PLSAV but missed by DBoW2 and CNN-based detector, including pure rotation, viewpoint variance, and moving objects

According to the table, only Mur-Artal [2], Khan [55], GPR [17], and PLSAV can achieve real-time performance for large-size image datasets, and the CNN-based method [33] has a competitive result with GPU support. GPR has the best performance of speed among all the algorithms, and PLSAV is just slightly behind because of the extra verification procedure. For large image resolution (M6L) the speed of the PLSAV achieves 80 fps, and the number increases to 543 for small image resolution (LI). There are four potential parts of acceleration in PLSAV. First, the image is divided into grids to perform the feature extraction and BoW vector generation in parallel with each grid small enough to reduce the computational complexity. Second, the histogram is computed simultaneously for a quick verification in the FLS without delay. Third, the task \( \mathcal{V} \) is running individually to correct the result at the back end as a verification rather than candidate retrieval (There is no need to compare with more previous frames). Finally, in the proposed LSC, when the state remains at “Initial Loop” or “Inside Loop” (state S1 or S2) rather than jumping to “No Loop” (state S0), the PLSAV does not search for the loop closure candidate from all the previous frames. In this case, only a few candidates (four in this paper) around last loop candidate are checked in parallel for each frame, which will benefit the loop closure detection system when the number of images in the sequence increases dramatically. Thus, the advantage of a multicore system is fully utilized to speed up the entire procedure.

The illustration of the acceleration of processing time on the dataset that has most images in Table 2 is shown in Figure 12, where we execute the original DBoW2 algorithm with ORB feature on our machine as well, which is the fastest algorithm in Table 5. It can be seen from the figure that PLSAV is not only faster than DBoW2 but more stable on execution time, which is more evident in a larger dataset like K00. The DBoW2 retrieves loop closure candidates from all the previous frames, thus the execution time varies widely. Actually, all the nearest-neighbour-based or BoW-based (tree-based) loop closure detection solutions will increase the processing time as the number of images grows. However, in PLSAV we only check four candidates in parallel for each frame at state S1 and S2, making the execution time stable most of the time. Besides, as the profile of time consuming is crucial for real-time localization applications, the execution time consumed per image on the CC dataset is also shown in Table 6, with each stage of PLSAV presented. The
FIGURE 12 Comparison of execution time between DBoW2 and PLSAV for every frame on (a) K00, (b) K05, (c) CC, and (d) M6L, the four datasets that contain the most images in Table 2.

TABLE 6 Execution time of each component in PLSAV on the CC dataset

| Component                                      | Mean ± STD of the execution time (ms/frame) |
|------------------------------------------------|---------------------------------------------|
| FLS Histogram                                 | 0.46 ± 0.10                                 |
| ORB feature extraction                        | 3.12 ± 0.92                                 |
| Weighted BoW and indexing                     | 2.97 ± 1.55                                 |
| LSC Querying (BoW or fastCandidate)           | 0.57 ± 2.82                                 |
| State control and decision                    | 0.35 ± 1.01                                 |
| Deep verification                             | 112.19 ± 7.17                               |
| Whole system                                  | 7.13 ± 2.98                                 |

FLS consists of the ORB-based BoW conversion and insertion, with a parallel process of histogram calculation and verification. The LSC will handle the candidates querying and state control. The DV task works in the background and returns the result after several cycles.

3.9 | CONCLUSION

A novel framework for real-time loop closure detection called PLSAV is presented in this paper, using the LSC to handle the continuous detecting procedure. This framework exploits a combination of a FLS task and a deep verification task in parallel structure, benefited by multithread computing on a multicore system. The FLS task divides images into small grids and calculates the BoW vectors together with histograms of the input images simultaneously. The deep verification task confirms the validity of the detection results with a lower frequency. The experimental results show that the execution time of the proposed framework achieves the best when compared with other state-of-the-art loop closure detection algorithm, providing real-time performance at a speed of 80–543 fps on a desktop with different kinds of image resolution, while the accuracy of the proposed framework is comparable. Future work will make attempts on other deep learning frameworks rather than CNN to determine the best architecture for loop closure application. Implementation on mobile platforms and acceleration using extra hardware like GPU and FPGA will be taken into consideration in the following work. Besides, the best parameters for PLSAV will be determined through Bayesian optimization among different datasets in future work rather than exhaustive searching over single dataset.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. This work was supported in part by the International Cooperation Research for Doctoral Students in Zhejiang University, the Zhejiang Provincial Natural Science Foundation of China (Grant No. LY19F020032), and the Zhejiang Provincial Key Research and Development Program of China (Grant No. 2021C03027).

ORCID

Zhe Yang https://orcid.org/0000-0001-7246-0012
Yun Pan https://orcid.org/0000-0002-9335-4291
Lai Deng https://orcid.org/0000-0002-5172-9411
Yuan Xie https://orcid.org/0000-0003-2093-1788
Ruojiang Huan https://orcid.org/0000-0003-2555-343X

REFERENCES

1. Liu, K., et al.: Visual place recognition via robust L2-norm distance based holism and landmark integration. AAAI 33(01), 8034–8041 (2019)
2. Mur-Artal, R., et al.: ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. IEEE Trans. Rob. 33(5), 1255–1262 (2017)
3. Newman, P., Cole, D., Ho, K.: Outdoor SLAM using visual appearance and laser ranging. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, pp. 1180–1187 (2006)
4. Mirovski, P., et al.: SignalSLAM: Simultaneous localization and mapping with mixed WiFi, Bluetooth, LTE and magnetic signals. In: International Conference on Indoor Positioning and Indoor Navigation, pp. 1–10 (2013)
5. Muresan, M.P., Nedevschi, S.: Multi-object tracking of 3D cuboids using aggregated features. In: 2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 11–18 (2019)
6. Xu, Y., et al.: How to train your deep multi-object tracker. arXiv:1906.06618 (2020)
7. Pang, J., et al.: Quasi-dense similarity learning for multiple object tracking. arXiv:2006.06664 (2020)
8. Yu, C., et al.: DS-SLAM: A semantic visual SLAM towards dynamic environments. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1168–1174 (2018)
9. Li, L., et al.: Dense 3D Semantic SLAM of traffic environment based on stereo vision. In: 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 965–970 (2018)
10. Galvez-López, D., Tardos, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Trans. Rob. 28(6), 1188–1197 (2012)
11. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-scale direct monocular SLAM. In: Computer Vision–ECCV 2014, pp. 834–849 Springer (2014)
12. Williams, B., et al.: A comparison of loop closing techniques in monocular SLAM. Rob. Auton. Syst. 57(12), 1188–1197 (2009)
13. Angeli, A., et al.: Fast and incremental method for loop-closure detection using bags of visual words. IEEE Trans. Rob. 24(5), 1027–1037 (2008)
14. Cummins, M., Newman, P.: FAB-MAP: Probabilistic localization and mapping in the space of Appearance. Int. J. Robot. Res. 27(6), 647–665 (2008)
15. Mur-Artal, R., Tardos, J.D.: Fast relocalisation and loop closing in keyframe-based SLAM. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), p. 846–853, IEEE, (2014)
16. Keijzer, N., Kumar, S., Shihata, T.: High performance loop closure detection using bag of word pairs. Robot Auton. Syst. 77, 55–65 (2016)
17. Yang, Z., et al.: Gridding place recognition for fast loop closure detection on mobile platforms. Electron. Lett. 55(17), 931–933 (2019)
18. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 225–234 (2007)
19. Fan, H., Ling, H.: Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. In: 2017 IEEE International Conference on Computer Vision (ICCV), p. 5487–5495,IEEE, Venice (2017)
20. Rublee, E., et al.: ORB: An efficient alternative to SIFT or SURF. In: 2011 International Conference on Computer Vision, pp. 2564–2571, IEEE (2011)
21. Zhang, H.: BoRF: Loop-closure detection with scale invariant visual features. In: 2011 IEEE International Conference on Robotics and Automation, pp. 3125–3130, IEEE, Shanghai (2011)
22. Shahbaz, H., Zhang, H.: Application of locality sensitive hashing to realtime loop closure detection. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1228–1233, San Francisco (2011)
23. Yin, J., Li, D., He, G.: Mobile robot loop closure detection using endpoint and line feature visual dictionary. In: 2017 2nd International Conference on Robotics and Automation Engineering (ICRAE), pp. 73–78 (2017)
24. Garcia-Fidalgo, E., Ortiz, A.: Hierarchical place recognition for topological mapping. IEEE Trans. Rob. 33(5), 1061–1074 (2017)
25. Milford, M., et al.: Condition-invariant, top-down visual place recognition. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 5571–5577, IEEE, Hong Kong (2014)
26. Nassef, T., Burgard, W., Stachniss, C.: Robust visual localization across seasons. IEEE Trans. Rob. 34(2), 289–302 (2018)
27. Neubert, P., Prozel, P.: Beyond holistic descriptors, keypoints, and fixed patches: Multiscale superpixel grids for place recognition in changing environments. IEEE Robot. Autom. Lett. 1(1), 484–491 (2016)
28. Cadena, C., et al.: Robust place recognition with stereo sequences. IEEE Trans. Rob. 28(4), 871–885 (2012)
29. Chen, J., et al.: A compact loop closure detection based on spatial partitioning. In: 2017 2nd International Conference on Image, Vision and Computing (ICIVC), pp. 371–375, IEEE, Chengdu (2017)
30. Garcia-Fidalgo, E., Ortiz, A.: iBoW-LCD: An appearance-based loop closure detection approach using incremental bags of binary words. IEEE Rob. Autom. Lett. 3(4), 3051–3057 (2018)
31. Chen, Z., et al.: Convolutional neural network-based place recognition. arXiv:1411.1509 (2014)
32. Sudderth, N., et al.: Place recognition with convexn landmarks: Viewpoint-robust, condition-robust, training-free. Robotics: Science and Systems XI, pp. 1–10 (2015)
33. Hou, Y., Zhang, H., Zhou, S.: Convolutional neural network-based image representation for visual loop closure detection. In: 2015 IEEE International Conference on Information and Automation, pp. 2238–2245, IEEE (2015)
34. Zhang, K., Zhang, W.: Loop closure detection based on generative adversarial networks for simultaneous localization and mapping systems. In: 2017 Chinese Automation Congress (CACC), pp. 7916–7919, Jinan (2017)
35. Kim, S., Ho, Y.: Place recognition for SLAM using texture-independent features and DCGAN. In: TENCON 2018—2018 IEEE Region 10 Conference, pp. 1269–1272, Jeju (2018)
36. Shin, D.-W., Ho, Y.-S., Kim, E.-S.: Loop closure detection in simultaneous localization and mapping using descriptor from generative adversarial network. JEI 28(1), 013014 (2019)
37. Arroyo, R., et al.: Towards life-long visual localization using an efficient matching of binary sequences from images. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 6328–6335, IEEE (2015)
38. Bampis, L., Ameraatidis, A., Gasteratos, A.: Encoding the description of image sequences: A two-layered pipeline for loop closure detection. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4530–4536, IEEE, Daejeon (2016)
39. Li, Y., et al.: Image sequence matching using both holistic and local features for loop closure detection. IEEE Access 5, 13835–13846 (2017)
40. Siam, S.M., Zhang, H.: Fast-SeqSLAM: A fast appearance based place recognition algorithm. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 5702–5708 (2017)
41. Kim, D.-H., Kim, J.-H.: Visual loop-closure detection method using average feature descriptors. In: Robot Intelligence Technology and Applications 2, Advances in Intelligent Systems and Computing, pp. 113–118, Springer, Cham (2014)
42. Bampis, I., et al.: A LoCAte-based visual place recognition system for mobile robotics and GPGPUs. Concurr. Comp. Pract. E 30(7), e4146 (2018)
43. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR’06), vol. 2, p. 2169–2178,IEEE, New York (2006)
44. Abadi, M., et al.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint, arXiv:1603.04467 (2016)
45. Zhou, B., et al.: Places: A 10 million image database for scene recognition. IEEE Trans. Pattern Anal. Mach. Intell. 40, 1452–1464 (2017)
46. Bianco, S., et al.: Benchmark analysis of representative deep neural network architectures. IEEE Access 6, 64270–64277 (2018)
47. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014)
48. Smith, M., et al.: The new college vision and laser data set. Int. J. Robot Res. 28(5), 595–599 (2009)
49. Blanco, J.L., et al.: A collection of outdoor robotic datasets with centimeter-accuracy ground truth. Auton. Robot. 27(4), 327 (2009)
50. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3354–3361 (2012)
51. Labbé, M., Michaud, F.: Appearance-based loop closure detection for online large-scale and long-term operation. IEEE T. Robot. 29(3), 734–745 (2013)
52. Zhang, X., et al.: Graph-based place recognition in image sequences with CNN features. J. Intell. Robot. Syst. 95(2), 389–403 (2019)
53. Vlaminck, M., Luong, H., Philips, W.: Have I seen this place before? A fast and robust loop detection and correction method for 3D LIDAR SLAM. Sensors 19(1), 23 (2018)
54. Ceriani, S., et al.: Rawseeds ground truth collection systems for indoor self-localization and mapping. Auton. Robot. 27(4), 353 (2009)
55. Khan, S., Wollherr, D.: IBuILD: Incremental bag of binary words for appearance based loop closure detection. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 5441–5447, IEEE (2015)