Hindawi
International Journal of Intelligent Systems
Volume 2024, Article ID 9992159, 14 pages
https://doi.org/10.1155/2024/9992159

WILEY | Hindawi

Research Article

Semisupervised Vector Quantization in Visual SLAM Using HGCN

Amir Zarringhalam,[1] **Saeid Shiry Ghidary**,[1,2] **Ali Mohades**,[1] **and Seyed-Ali Sadegh-Zadeh**[3]

[1]*Amirkabir University of Technology, Computer Science and Mathematics, 424 Hafez Ave, Tehran, Iran*
[2]*Staffordshire University, School of Digital, Technologies and Arts, College Rd, Stoke-on-Trent ST4 2DE, UK*
[3]*Staffordshire University, Department of Computing, Stoke-on-Trent, UK*

Correspondence should be addressed to Saeid Shiry Ghidary; saeed.shiryghidary@staffs.ac.uk

Received 26 March 2023; Revised 13 January 2024; Accepted 17 January 2024; Published 29 February 2024

Academic Editor: Alexander Hošovský

We present a novel vector quantization (VQ) module for the two state-of-the-art long-range simultaneous localization and mapping (SLAM) algorithms. The VQ task in SLAM is generally performed using unsupervised methods. We provide an alternative approach trough embedding a semisupervised hyperbolic graph convolutional neural network (HGCN) in the VQ step of the SLAM processes. The SLAM platforms we have utilized for this purpose are fast appearance-based mapping (FABMAP) and oriented fast and rotated short (ORB), both of which rely on extracting the features of the captured images in their loop closure detection (LCD) module. For the first time, we have considered the space formed by these SURF features, robust image descriptors, as a graph, enabling us to apply an HGCN in the VQ section which results in an improved LCD performance. The HGCN vector quantizes the SURF feature space, leading to a bag-of-word (BoW) representation construction of the images. This representation is subsequently used to determine LCD accuracy and recall. Our approaches in this study are referred to as HGCN-FABMAP and HGCN-ORB. The main advantage of using HGCN in the LCD section is that it scales linearly when the features are accumulated. The benchmarking experiments show the superiority of our methods in terms of both trajectory generation accuracy in small-scale paths and LCD accuracy and recall for large-scale problems.

1. Introduction

Autonomous robots are increasingly utilized in various fields to assist humans with complex tasks. Regardless of the domain, it is crucial for robots to be able to operate safely and robustly in dynamic and unfamiliar environments [1]. The emergence of autonomous robotics along with the integration into the principles of industry has become a prevalent topic, primarily due to technological advancements [2]. Autonomous robots and vehicles that possess the ability to navigate complex environments over prolonged time intervals are desirable in numerous applications.

Accurate localization is crucial in autonomous robots, as it is necessary for a robot to navigate precisely and reach its designated goal locations [3]. In many situations, robots have no prior knowledge of the environment and no global positioning system (GPS) is available. Although GPS provides acceptable localization accuracy, in various situations, such as indoor places, tunnels, and even urban areas where buildings obscure GPS signals, it may fail to provide proper localization, causing inconsistence performance of robots. Visual simultaneous localization and mapping (vSLAM) provides an alternative approach to address the navigation challenges in unknown areas with no accessible GPS. vSLAM enables autonomous vehicles to construct maps and navigate properly using only visual sensors like cameras. In vSLAM, place recognition is defined by the loop closure detection (LCD) problem, which refers to the process of identifying a location where the robot or the device has previously visited a loop.

Trajectory generation and LCD are vital for preserving localization accuracy, and they constitute the core part of every SLAM process. While many solutions to the SLAM

problem have been proposed, several challenges like loop closure detection in a long range remain to be addressed. Approaches like FABMAP had great success in several large-scale experiments, including the 1000 km trajectory planning in England [1]. However, this method suffers from low LCD recall.

In this research, we provide a solution for vector quantization (VQ) by envisioning feature space representation in another way. First, the extracted SURF features from images are encoded as a graph, which enables us to apply a graph convolutional neural network (GCN) for feature classification. Second, we introduce a new semisupervised node prediction algorithm that replaces the custom unsupervised clustering algorithm as a part of VQ. Accuracy and recall of the LCD are enhanced significantly as a consequence of employing the new VQ step in the FABMAP2 algorithm.

Though not proven, we conjecture that large-scale SLAM performance degradation in FABMAP2 is caused by incorrect clustering on the SURF feature space. The intuition behind this assumption is that features are visualized as a tree with numerous potential cycles, and as the images accumulate, the number of features grows exponentially and becomes intertwined. Hence, Euclidean embedding may not be suitable for such complex graph structures. The inadequacy of Euclidean spaces stems from the fact that they cannot represent the distances between the knotted features properly. Thus, clustering algorithms like Kmeans, Kmeans++, and partition around medoids (PAM) which employ pairwise Euclidean distances between data points fail to vector quantize the space properly. Hence, we propose using a hyperbolic embedding that is suitable for our intricate hierarchically structured graph feature data. Next, the VQ module in the FABMAP2 algorithm that uses BoW representation of images is replaced with a hyperbolic-based vector quantization module. The computational complexity of BoW-based SLAM is not significantly higher than that of other methods, and it is feasible to use this technique as a standard method in the long-range SLAM algorithm's LCD section. The main disadvantage of BoW-based loop closure detection on large-scale problems is the increment in vocabulary size for the maps that are built from a large number of images. Accordingly, the VQ process, which involves matching thousands of features against each other, becomes expensive compared to small-scale SLAM. Moreover, there is also an associated low recall due to perceptual aliasing.

In our approach, we rebuild the vocabulary and replace the BoW construction block (normally produced from unsupervised algorithms) with a semisupervised approach called HGCN [4]. We integrate this model in two SLAM algorithms, ORB-SLAM and FABMAP2, to solve the LCD problem. Our benchmark results show that the new FABMAP algorithm (which we call HGCN-FABMAP) outperforms the original FABAMAP2.

The paper is organized as follows: Section 2 contains a review of some of the current BoW-based LCDs in the SLAM method literature. In Section 3, a concise background on FABMAP2, ORB-SLAM, and HGCN is presented. Furthermore, a comprehensive study of our novel methodologies is presented in this section. Section 4 contains experiments and results. Finally, Section 5 contains the future direction of our research.

2. Related Works

In this section, we briefly review the BoW-based LCD detection methods, but we avoid deeply delving into them so as not to get distracted from presenting our main study. We have prepared a nutshell and a brief background of our main methods in the next section.

To the best of our knowledge, no supervised or semisupervised methods currently address vector quantization in SLAM.

In [5, 6], the authors proposed a stochastic graph nearest neighbor (SGNN) search to accelerate BoW image retrieval. SGNN starts at a random node and uses a greedy approach to traverse the graph structure. The computational cost of finding an approximate nearest neighbor for a random query node Q in SGNN is bounded by $\min\{n^d, 2n^{1/d}d^2\}$, where n is the number of points and d is the dimension of the hypercube (volume 1) from which the points are randomly selected. This technique is highly effective; nevertheless, it imposes strict restrictions on the distance between each point and its closest neighbor.

To overcome regular BoW limitations such as perceptual aliasing, Kejriwal, Kumar, and Shibata proposed an online method to construct a bag-of-word pair (BoWP) approach which uses spatial co-occurrence of words [7].

Han et al. in [8] proposed an incremental method for learning and updating binary vocabulary which is invariant to robot pose, and it is suitable for loop closure detection as it utilizes a simplified likelihood function tailored to this objective. The generation process of binary vocabulary is based on tracking features across consecutive images.

To mention other methods, Garcia-Fidalgo et al. present iBoW-LCD, a novel online-like appearance-based LCD method that employs an incremental BoW schema on feature descriptors, eliminating the necessity for a predetermined training vocabulary [9]. iBoW-LCD makes use of an efficient mechanism to group images similar in time complexity and associated with binary feature words [9].

Maxime Ferrera [10] proposes OV^2, as another online visual SLAM algorithm that handles the SLAM tasks concurrently using a multithreaded system. Lucas–Kanade optical flow has been employed to track and provide camera pose estimations. To provide continuous localization through 3D map population and drift minimization in a single local map-tracking step, OV^2 makes use of iBoW-LCD [9] for loop closing.

3. Methods

3.1. Backgrounds

3.1.1. FABMAP2. In this section, we try to present the previous works in which our main methods are based on.

In [11], two versions of the FABMAP algorithm are introduced that make use of a probabilistic framework for

environments with distinct locations. The LCD module of both versions uses a BoW representation, and at time k, the map is built up using locations $\mathscr{L}^k = \{L_1, \ldots, L_{n_k}\}$. Here, each L_i is the set $\{p(e_i = 1|L_i), \ldots., p(e_{|\nu|} = 1|L_i)\}$, where $e_i s$ correspond to scene elements generated by the locations. In FABMAP2, local scene observations, Z_k, is the set $\{z_1, \ldots, z_{|\nu|}\}$, where z_i is a binary variable indicating the presence or absence of the i^{th} word in the vocabulary. For environments with more than hundreds of thousands of locations in the map, an inverted index technique needs to organize visual words, aiding FABMAP2 in large dataset loop closure detection, and we avoid using FABMAP1. This is because it needs the computation of appearance likelihood (observation) factors $\prod_{(q=2)}^{(|\nu|)} p(z_q \mid z_{p_q}, L_i)$, where z_q is the children of z_{p_q} in the Chow–Liu tree, and this is intractable. A modification in the parameterized model of each location makes the computation of $p(z_q \mid z_{p_q}, L_i) = \sum_{s_{e_q} \in \{0,1\}} p(z_q \mid e_q = s_{e_q}, z_{p_q}) p(e_q = s_{e_q} \mid L_i)$. To illustrate more, suppose that our map has only 4 locations, the FABMAP1 method takes different values for the term $p(z_q \mid z_{p_q}, L_i)$ for each location (this is an unrestricted model). For the restricted model (FABMAP2), locations share the same value for the term $p(z_q \mid z_{p_q}, L_i)$ when q is not observed, as indicated in Table 1.

3.1.2. ORB-SLAM. On the line of our research, we make use of ORB-SLAM to present its modified version.

ORB-SLAM [12] is built on the primary ideas of Klein and Murray's PTAM [13]. In [12], instead of limiting the vSLAM procedure to incremental mapping frame-by-frame that results in a sparse map with high-quality features, a solution with more dense features but lower quality is proposed. The PTAM algorithm has several drawbacks; for example, it is only applicable in small-scale environments. Further disadvantages of this algorithm include an inefficient loop closing method and the requirement for human input for map bootstrap. ORB-SLAM as a monocular SLAM method has overcome the drawbacks of the PTAM's algorithm as discussed in the introduction section of [12].

This algorithm establishes initial correspondences by extracting features from the current frame F_c and searches for matches between these features and those in the reference frame F_r. Next, two parallel threads are initiated to compute geometric models for both planar and nonplanar scenes:

$$\begin{aligned} x_c &= H_{cr} x_r, \\ x_c F_{cr} x_r &= 0. \end{aligned} \tag{1}$$

Here, x_c is the current frame, x_r is the reference frame, and H_{cr} and F_{cr} are homography and fundamental matrices. Next, as part of a RANSAC scheme, the procedure for both models is made homogeneous by predefining the number of iterations and the points to be used at each one. That is, 8 points are used for the F_{cr} matrix and 4 points for the H_{cr} [14]. At each iteration, a score SM is computed for each model M:

$$S_M = \sum_i \rho_M\left(d_{cr}^2\left(x_c^i, x_r^i, M\right)\right) + \rho_M\left(d_{rc}^2\left(x_c^i, x_r^i, M\right)\right), \tag{2}$$

$$\rho_M\left(d^2\right) = \begin{cases} \Gamma - d^2, & if \quad d^2 < T_M, \\ 0, & if \quad d^2 \geq T_M, \end{cases} \tag{3}$$

where d_{cr}^2 and d_{rc}^2 in formula (2) are symmetric transfer errors from one frame to another and T_M in (3) is the outlier rejection threshold. To ensure homogeneity in the process, Γ equals T_H which is a user-defined threshold. This results in both models having the same score for the same d in their inlier region. F_{cr} and H_{cr} are matrices with the highest score retained. In steps three and four, a heuristic method determines whether to use H_{cr} or F_{cr}. By employing H_{cr}, the model gets initialized from the plane. In situations with high parallax, the heuristic of utilizing F_{cr} is adopted:

$$R_H = \frac{S_H}{S_H + S_F}. \tag{4}$$

If $R_H > 0.45$ homography matrix is used and otherwise the fundamental is selected.

3.1.3. HGCN. Our secondary aim is to investigate HGCNs into the previously discussed SLAM methods.

In [15], the authors introduced graph convolutional neural networks (GCNs), a variant type of convolutional neural network, which directly operate on graphs. This model scales linearly with the number of graph edges. However, dealing with graphs with exponentially many leaves might not respect the structures very well.

Representing data in spaces such as hyperbolic or spherical has been growing due to their occurrence in real-world scenarios. For example, [4, 16] extend the capabilities of graph neural networks to better handle data with non-Euclidean properties. HGCN [4] resolves difficulties associated with unsupervised methods including large distortion in embedding real-world graphs. Here, it is shown how HGCN works, and at the end of this section, two more methods are further discussed.

In HGCN, information propagation consists of three steps: message computation, aggregation, and update.

In the message computation step, neighboring nodes exchange messages based on relationships between data points:

$$* h_i^{l,H} = \left(W^l \otimes^{K_{l-1}} x_i^{l-1,H}\right) \oplus^{K_{l-1}} b_l = \mathrm{MSG}\left(x_i^{l-1,H}\right). \tag{5}$$

The two operators $\otimes$ and $\oplus$, namely, linear (conventional matrix multiplication) and Mobius addition, are formulated for hyperbolic spaces as

$$\begin{aligned} W \otimes^k X^{l,H} &= \exp_0^k\left(W \log_0^k\left(X^{l,H}\right)\right), \\ X^H \otimes^k b &= \exp_{X^H}^k \exp\left(P_{o \longrightarrow X^H}^k (b)\right). \end{aligned} \tag{6}$$

Table 1: Exploring spatial models: unrestricted model with unique location values, FABMAP2's log-likelihood consistency for unobserved "q" values, and the normalized interpretation of FABMAP2.

	L_1	L_2	L_3	L_4
a	$p(z_q \mid z_{p_q}, L_1)$	$p(z_q \mid z_{p_q}, L_2)$	$p(z_q \mid z_{p_q}, L_3)$	$p(z_q \mid z_{p_q}, L_4)$
b	$p(z_q \mid z_{p_q}, L_1)$	$p(z_q \mid z_{p_q}, L_2)\mid_0$	$p(z_q \mid z_{p_q}, L_3)\mid_0$	$p(z_q \mid z_{p_q}, L_4)$
c	$\log(p(z_q \mid z_{p_q}, L_1)/p(z_q \mid z_{p_q}, L)\mid_0)$	0	0	$\log(p(z_q \mid z_{p_q}, L_4)/p(z_q \mid z_{p_q}, L)\mid_0)$

Part (a) depicts an unrestricted model where each location can take a specific value. In part (b), the FABMAP2 model is shown in which log-likelihood of locations where "q" values are not observed must take the same value. Part (c) shows the normalized version of part (b) [1].

To aggregate neighbor messages for hidden representation computation, the data points need to be mapped to a tangent space to the hyperbolic space. Next, central gravity is calculated, and the points are mapped back to the next hyperbolic layer:

$$* y_i^{l,H} = \mathrm{AGG}^{k_{l-1}} h_i^{l,H}. \tag{7}$$

In update function formula (8), a nonlinearity is applied and mapped back to a different curvature. Notably, the inverse curvature of the layers l and $l-1$ is represented as k^l and k^{l-1}. The mapping between hyperbolic and Euclidean tangent space is also performed:

$$x_i^{l,H} = \sigma^{\otimes^{k_{l-1},k_1}} y_i^{(l,H)} = \mathrm{Update}^{k_{l-1},k_l}\left(y_i^{l,H}\right), \tag{8}$$

where

$$\sigma^{\otimes^{k_{l-1},k_1}}\left(X^H\right) = \exp_o^{k_l} \quad \left(\sigma\left(\log_o^{k_{l-1}}\left(X^H\right)\right)\right). \tag{9}$$

The underlying principle of using hyperbolic spaces is the existence of exponentially many points in Euclidean spaces. Node features in Euclidean spaces are learned by most GCNs [17]. However, it has been shown that hyperbolic geometry provides more ability to embed graphs with scale-free or hierarchical structures compared with Euclidean geometry [4, 17, 18]. HGCN [4] extends the graph convolution on the hyperboloid manifold of the hyperbolic spaces, and feature transformation is actually conducted in tangent spaces [18]. The hyperbolic graph operations, including feature transformation and nonlinearity activation, are derived from the Lorentz GCN (LGCN) framework to ensure the transformed node features follow hyperbolic geometry [18]. Our primary contribution in this work is to present and substitute a new hyperbolic vector quantization module in the FABMAP2 algorithm, which we will discuss in the next section.

3.2. New Methods

3.2.1. HGCN-Vector Quantization. Vector quantization is a procedure used in computer vision to assign image features to their nearest corresponding cluster centroids. In our approach, we first classify the extracted SURF feature space using a HGCN. Next, using the clustering centroids, we calculate the quantized version of the vectors by

$$Q_k^{(i)} = \left\{\underline{x} : \left|\underline{x} - \hat{\underline{x}}_k^i\right| \leqslant \left|\underline{x} - \hat{\underline{x}}_k^j\right|; \quad \forall j \in \{1, \ldots, n\}\right\}. \tag{10}$$

3.2.2. HGCN-FABMAP. The main part of the LCD section in the FABMAP2 process uses BoW representation for Chow–Liu tree training. Furthermore, BoW representation construction requires finding clustering centroids (vocabulary) of the feature data, which up to now has been carried out using unsupervised approaches.

Here, we introduce a semisupervised approach with HGCN for this step. The reason for such selection is that extracted SURF features from images form a tree with a large number of interleaved leaves, which causes distortion. To make the features more clusterable, we use HGCN to map the features to a hyperboloid, hence reducing distortion and increasing the LCD accuracy. One of the primary advantages of using HGCN over traditional unsupervised methods is that the number of extracted features scales linearly. Our method, HGCN-FABMAP, is built upon FABMAP2.

In the initial version of FABMAP (FABMAP1; described in the appendix), locations are not constrained, and the log-likelihood of each location can take any value. However, as presented in Table 1, in FABMAP2, various constraints are applied to the likelihood values of places, see Section 3.1.1. The likelihood of locations where a visual word q has not been seen before has a fixed value. Furthermore, the likelihoods can be incremented using one of the following values for a given word and a particular location (see appendix):

$$\begin{cases} \mathrm{Case}_1\colon \left(S_q = 1, S_{p_q} = 1\right), \\ \mathrm{Case}_2\colon \left(S_q = 1, S_{p_q} = 0\right), \\ \mathrm{Case}_3\colon \left(S_q = 0, S_{p_q} = 1\right), \\ \mathrm{Case}_4\colon \left(S_q = 0, S_{p_q} = 0\right). \end{cases} \tag{11}$$

As discussed in Section 3.1.1, $z_i \in Z_k = \left\{z_1, \ldots, z_{|v|}\right\}$ represents a noisy measurement of visual words q. States in formula (13) are determined by the presence or the absence of the visual word q and its parent p_q. Because observations are typically sparse, and since the inverted-index technique is employed, Case4 in which both z_q and z_{p_q} are zero is the most likely scenario for our observations. Thus, the default

likelihood for each location L_i is computed using Case4, as shown in Algorithm 1.

In this algorithm:

$$d_1^* = \frac{((1 - \text{CLtree}(1,q)) \times 0.61 \times (1 - \text{CLtree}(3,q)))}{(\text{CLtree}(1,q) \times 0.39 \times (1 - \text{CLtree}(3,q)) + (1 - \text{CLtree}(1,q)1,q) \times 0.61 \times (1 - \text{CLtree}(3,q)))}, \tag{12}$$

$$d_1 = \log(d_1^*). \tag{13}$$

In (13), d_1 is the initial likelihood for each location, and it is set to a default value that assumes a null observation; i.e., q has occurred neither in the observation nor in the observation's parent. This value can be viewed as the sum of default votes for each observed word at the location. It should be noted that the default likelihood will vary for each location. This first algorithm is responsible for initializing the locations to their default likelihood.

The HGCN-FABMAP method, like the FABMAP2 algorithm, returns a confusion matrix using which the LCD accuracy and recall are calculated. In formula (12), q represents the index of the word in the vocabulary for each image and ranges from 0 to $|\mathscr{C}|$, and CLtree (i,q) represents the value of the Chow–Liu tree at level i for the index q. While processing new observations, we only need to adjust the default likelihood of locations. This adjustment is represented in Algorithm 2. In the first part of Algorithm 2, we only evaluate the changes where $z_q = 1$.

Weights will be updated according to the content of the current observation using the terms d_3 and d_4. In the final section of this algorithm, the likelihood is altered when the word q is not observed and its parent in the Chow–Liu tree is present:

$$d_{2_{\text{num}}} = \frac{(\text{CLtree}(1,q) \times 0.61 \times (1 - \text{CLtree}(2,q)))}{((1 - \text{CLtree}(1,q)) \times 0.39 \times \text{CLtree}(2,q))}, \tag{14}$$

$$d_{2_{\text{den}}} = \frac{\left(\text{CLtree}(1,q) \times 0.61 \times \text{CLtree}(2,q)^2 \times 0.39\right)}{([(1 - \text{CLtree}(1,q)) \times 0.61 \times (1 - \text{CLtree}(2,q)) + \text{CLtree}(1,q) \times 0.39 \times \text{CLtree}(2,q)] \times (1 - 0.39 \times \text{CLtree}(1,q)))}, \tag{15}$$

$$d_2 = \log\left(\frac{d_{2_{\text{num}}}}{\left(1 - d_{2_{\text{den}}}\right)}\right) - d_q, \tag{16}$$

$$d_{3_{\text{den}}} = \frac{((\text{CLtree}(1,q)) \times (1 - \text{CLtree}(1,q))0.39 \times \text{CLtree}(3,q))}{((1 - \text{CLtree}(1,q)) \times \text{CLtree}(1,q) \times (1 - \text{CLtree}(3,q)))}, \tag{17}$$

$$d_{3_{num}} = \frac{((1 - \text{CLtree}(1,q)) \times 0.39 \times \text{CLtree}(3,q))}{((1 - \text{CLtree}(1,q)) \times 0.39 \times \text{CLtree}(3,q) + \text{CLtree}(1,q) \times 0.61 \times (1 - \text{CLtree}(3,q)))}, \tag{18}$$

$$d_3 = \log\left(\frac{d_{3_{\text{num}}}}{d_{3_{\text{den}}}}\right) - d_q, \tag{19}$$

$$d_4^* = \frac{(\text{CLtree}(1,q) \times 0.61)}{(1 - 0.39 \times \text{CLtree}(1,q))}, \tag{20}$$

$$d_4 = \log(d_4^*) - d_q, \tag{21}$$

d_3 and d_4 in formulas (21) and (23) are used to calculate the log-likelihood of the locations where word q was observed. d_2 in formula (18) is used to calculate the log-likelihood of unobserved words that are children of observed parent words in CLtree. Next, to update the log-likelihood of each of these locations, the default vote D_q is subtracted and the appropriate vote is added.

In equations (14), (16), (17), (23), and (22), CLtree refers to the Chow–Liu tree created from the training vocabulary using the BoW representation, which is obtained by applying

```
(1) For q ← 1 to #clusters do
(2)     Locations ← Inverted-Index [q]
(3)     For L_i in Locations do
(4)         Log-Likelihood [L_i] += d_1
(5)     End for
(6) End for
```

Algorithm 1: Default log-likelihood.

```
(1) For z_q ∈ Z where z_q = 1 do
(2)     Locations ← Inverted-Index [q]
(3)     For L_i in Locations do
(4)         If CLtree (0, 1) > 0 then
(5)             Log-Likelihood [L_i] += d_4 − d_1
(6)         else
(7)             Log-Likelihood [L_i] += d_3 − d_1
(8)         End if
(9)     End for
(10) End for
(11) For z_q ∈ Z where z_q = 0 and z_{p_q} = 1 do
(12)    Locations ← Inverted-Index [q]
(13)        For L_i in Locations do
(14)            Log-Likelihood [L_i] += d_2 − d_1
(15)        End for
(16) End for
```

Algorithm 2: Filling log-likelihood.

HGCN on the image's feature space. For more information about the formulation, we refer the reader to the source code in [9].

When applying HGCN, we need to calculate N-connected nodes for each feature in the space. For this process, we utilize the parallel Faiss algorithm described in [19] to obtain the nearest connected nodes.

For the modified FABMAP2 method, after extracting features for both training and test data, the BoW representation of images is calculated. As shown in Figure 1, the set of all features is obtained for each image and vector quantized on the cluster centroids of the features extracted from all train/test images. Following that, TF-IDF weighting is applied to the image's BoW representation. This procedure is repeated for each image, resulting in a BoW representation of all images. Next, a Chow-Liu tree is trained using only the training data, their cluster centroids, and training BoW data. Using the obtained CLtree and BoW representation of image i, we determine whether a new place is detected, or we have encountered a loop closure. At the end of the algorithm, a confusion matrix is outputted through similarity between images; therefore, potential loop closures can be obtained.

FABMAP2 and its HGCN-extended version require no repetitive paths in the training and testing datasets. The two datasets must also have similar contexts; otherwise, the algorithm will not perform well. For all the algorithms tested in this article, we have extracted the standard 128-d SURF features from the dataset. The feature matrix is then projected onto the PCA coordinates (with the number of principal components set to 20), which is then inputted into the Faiss algorithm [19] to construct the adjacency graph. The number of the closest node is set to 9, and this number is obtained by trial and error and sufficiently describes the graph. Next, we feed the obtained data into a 2-layer hyperbolic graph convolutional neural network implemented in PyTorch [4] to obtain cluster centroids of the images. BoW representation is then obtained in the next step and along with the centroids will be used in the HGCN-FABMAP procedure.

Next, as shown in Figure 1, the sequence of images in the test dataset is read one by one, and the new place likelihood is calculated. Based on the data association rule, the algorithm determines e to update and add the new location in the map or to declare loop closure.

The described procedure, HGCN-FABMAP, is depicted in Figure 1, and it is implemented using the modified C++ package provided by [9].

3.2.3. HGCN-ORB SLAM. HGCN-ORB is a SLAM algorithm. We assume that oriented fAST and rotated brief (ORB) features extracted from a sequence of images form a graph or tree with a large number of leaves. ORB-HGCN SLAM in the LCD section in the offline phase uses a new BoW representation of all images, which is created by applying an HGCN over the extracted features. In the online phase, the BoW representation is incorporated into the LCD section. Here, we replace the obtained LCD module using

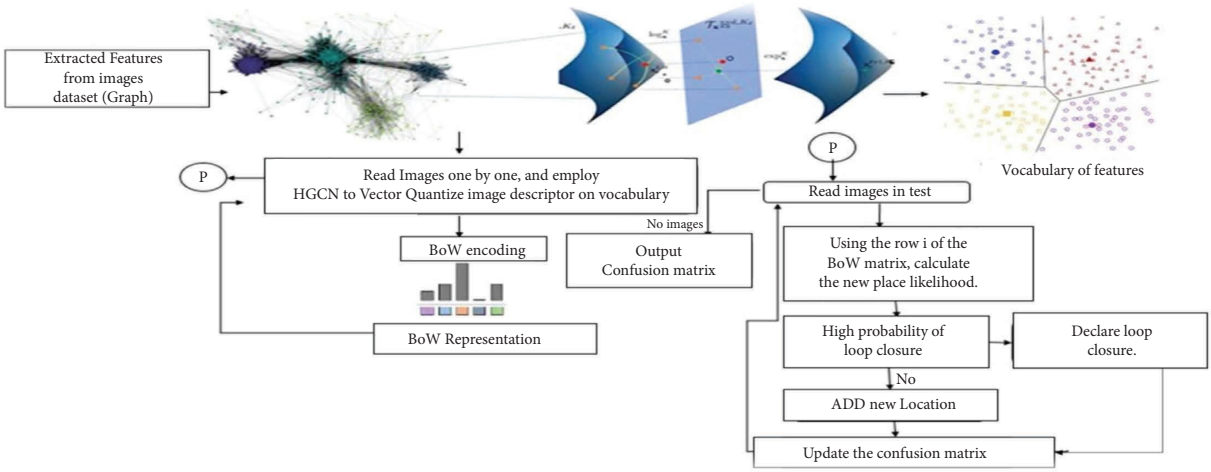


Figure 1: The overall HGCN-FABMAP procedure.

hyperbolic GCN in the ORB-SLAM trajectory generation process and compare the root mean squared error (RMSE) of the plotted trajectory against the ground truth trajectory. Here, the path is the surrounding area of a table which is approximately a circle.

4. Experimental Results

We utilized both indoor and outdoor datasets to evaluate the performance of our methods, HGCN-FABMAP, HGCN-BoW, and HGCN-ORB. Our presented methods are benchmarked against FABMAP, BoW, ORB, LSD-SLAM, ORB-SLAM2, FOVIS, and methods described in references [20, 21] and [22].

Performance evaluations were carried out on TUM sequence 11, St. Lucia (train/test), Newer College (short experiment), New College, and Lip6indoor datasets. ORB-SLAM was also applied to the TUM Freiburg-3 long office household dataset. The datasets used in this study are briefly reviewed in the following section. The ORB algorithm has only been tested on the Frieburg-3 dataset.

4.1. Datasets Used for HGCN-FABMAP and HGCN-BoW. Table 2 summarizes the information about the source, number of images, and the environment for each dataset. Sample images from each of the datasets in Table 2 are depicted in Figure 2.

Figure 2 represents samples of images from each dataset. Part (a) of this figure represents part of TUM dataset sequence-11, parts (b) and (c) represent part of the Oxford university dataset, part (d) is the Lip6indoor dataset which is a narrow corridor of a lab, and the last part is the Frieburg-3 dataset.

4.2. Loop Closure Detection Performance Metrics. For model performance evaluation, we use the same metrics as in [13]. The recall and accuracy of loop closure detection are defined by

$$\text{recall} = \frac{\sum_i \sum_j ((\text{ConfusionMatrix}[i][j] > \text{threshold}) \wedge (\text{groundtruth}[i][j] == 1))}{\sum_i \sum_j (\text{ConfusionMatrix}[i][j] == 1 > \text{threshold})}, \tag{22}$$

$$\text{accuracy} = \frac{\sum_i \sum_j ((\text{ConfusionMatrix}[i][j] > \text{threshold}) \wedge (\text{groundtruth}[i][j] == 1))}{\sum_i \sum_j (\text{groundtruth}[i][j] == 1)}. \tag{23}$$

The HGCN-FABMAP algorithm outputs a confusion matrix in which rows and columns represent images, and each entry (i^{th}, j^{th}) represents a number between zero and one (more similar).

The numerator in formulas (22) and (23) indicates true positives, i.e., the number of elements in the confusion matrix greater than a given threshold for which the corresponding elements in the ground truth matrix are also ones.

Table 2: An overview of the datasets utilized in various studies, including HGCN-FABMAP, HGCN-BoW, HGCN-ORB College, Lip6indoor, and Frieburg3.

Datasets	Number of images		Long range	Dataset description	Ref.
	Train	Test			
Lip6indoor dataset	350			Images are taken from a lab environment with a narrow corridor	[23]
TUM sequence-11	1500			Images are taken from a lab environment with different lighting conditions. These data are our training dataset, where Lip6indoor is the test dataset	[24]
New College	1073		●	This dataset has been taken from the Oxford university campus, which includes complex repetitive structures. This is a stereo dataset with left and right sequences. We have used the left sequence containing images with 640×480 resolution as our training data for the Newer College dataset	[25]
Newer College	200			This is a large video with the same context of the New College dataset, and it contains 3 loops. The camera is experiencing cluttered movements	[26]
St. Lucia suburbs	540	1000	●	Images are taken by a camera-equipped car. The dataset has two parts: a training dataset and a set of test images as test data	[27]
Freiburg-3	2500			Images taken from the surrounding area of a table and it contains one loop	[28]

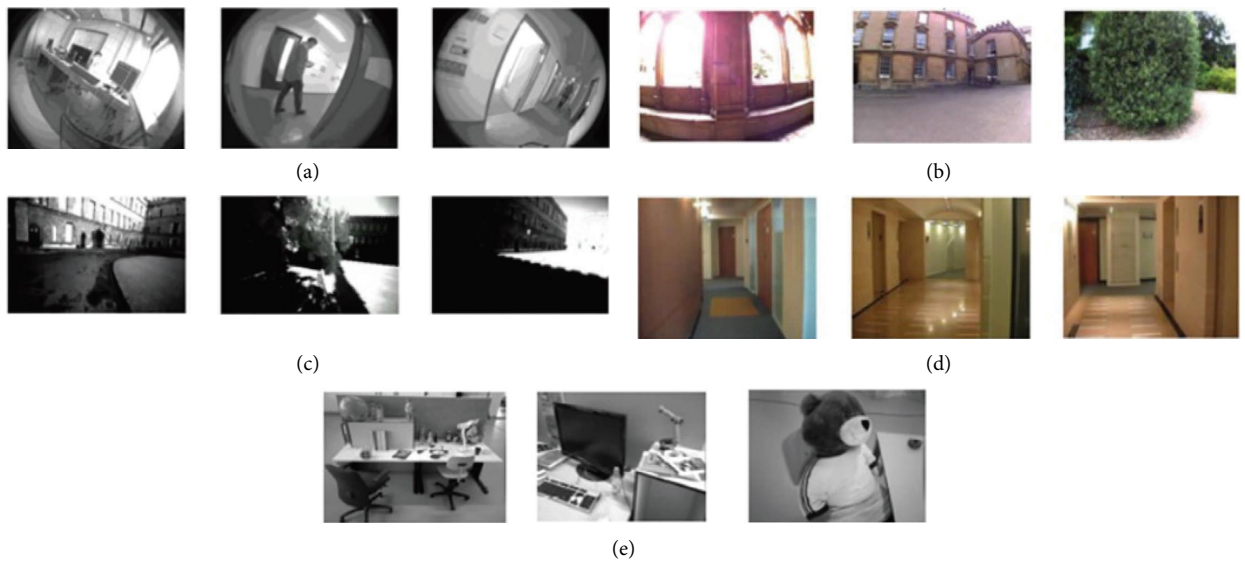


Figure 2: Example images of the used datasets. (a) TUM sequence 11, (b) New College, (c) Newer College, (d) Lip6indoor, and (e) Frieburg-3 datasets.

Training and testing data in benchmarking Tables 3 and 4 are indicated in columns 1 and 2, respectively. Accuracy and recall in each experiment are reported for HGCN-FABMAP and FABMAP2 in Table 3.

Table 4 shows the same measurement for HGCN-BoW and BoW methods. The accuracy and recall of the algorithm that performs the best are represented in bold. As can be seen, our algorithm performs better than FABMAP2 and BoW in all cases. The first row of Table 3 demonstrates that HGCN-FABMAP outperforms the normal FABMAP2 algorithm for the New College dataset, with St. Lucia (train) dataset serving as our training data. The second row of Table 3 displays the results of comparing the two methods HGCN-FABMAP and FABMAP2, with TUM sequence 11 as our training dataset and Lip6indoor lab as our testing dataset. FABMAP2 and HGCN-FABMAP perform similarly here, with the exception of recall 81.5, where HGCN-FABMAP performs somewhat better. Our conjecture behind this observation is that the environment of TUM sequence 11 is not a perfect match for the Lip6indoor dataset. In addition, CLtree was formed on a very limited training set. The third row in Table 3 shows that FABMAP2 and HGCN-FABMAP perform equally well on the Newer College dataset.

The confusion matrices represented in Figure 3 have a fuzzy nature (instead of binary) as is the case in FABMAP2. Each element of the confusion matrices plotted in Figure 3 is the dot product between the two images i and j weighted by the TF-IDF representation. Dark regions represent higher similarity or potential loop closures. Figures 3(b–d) depict the confusion matrix produced by applying HGCN-BoW to the Lip6indoor, Newer College, and St. Lucia datasets.

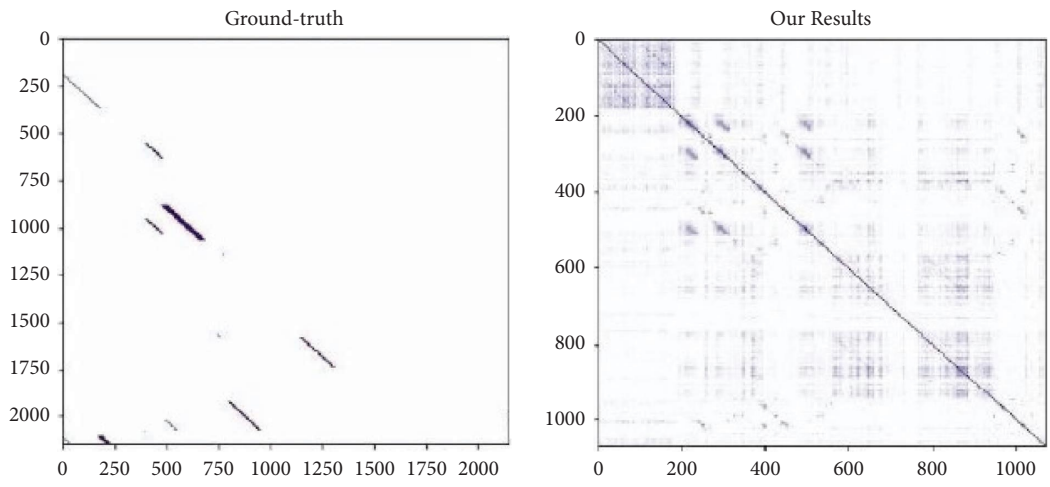
Table 4 represents the comparison between HGCN-BoW and regular BoW algorithm. To determine the accuracy and recall of loop closure detection, a ground truth confusion matrix is required. Typically, a binary matrix that represents the ground truth is used for this purpose, and it is only

Table 3: Comparing HGCN-FABMAP and FABMAP2 methods for LCD.

Train dataset	Test dataset	HGCN-FABMAP		FABMAP2	
		Acc (%)	Rec (%)	Acc (%)	Rec (%)
St. Lucia	New College	50.07	100	50.02	100
311417	160500	56.00	86.00	56.8	82.00
		77.30	68.62	77.60	63.00
TUM sequence-11	Lip6indoor	50.30	100	50.34	100
392449	27189	54.90	90.50	54.90	90.50
		59.55	81.50	52.59	81.20
St. Lucia	St. Lucia	75.00	72.00	50.80	100
311417	235510	54.05	95.85	50.40	90.00
				59.55	81.50
New College	Newer College	90.20	80.85	71.00	80.30
160500	194000	97.40	72.34	79.00	72.34
		99.00	67.75	80.02	67.55

Table 4: Comparing HGCN-BOW and BoW methods for LCD.

		HGCN-BoW		BoW	
		Acc (%)	Rec (%)	Acc (%)	Rec (%)
St. Lucia	New College	50.00	100	50.00	85.00
311417	160500	91.00	70.50		
TUM sequence-11	Lip6indoor	50.03	100	34.00	100
392449	27189	54.48	90.00		
		60.00	81.00		
St. Lucia	St. Lucia	62.00	72.00	34	72.00
311417	235510	58.90	99.00		
New College	Newer College	Ground	Ground	44.00	100
160500	194000	Truth	Truth		



(a)

Figure 3: Continued.

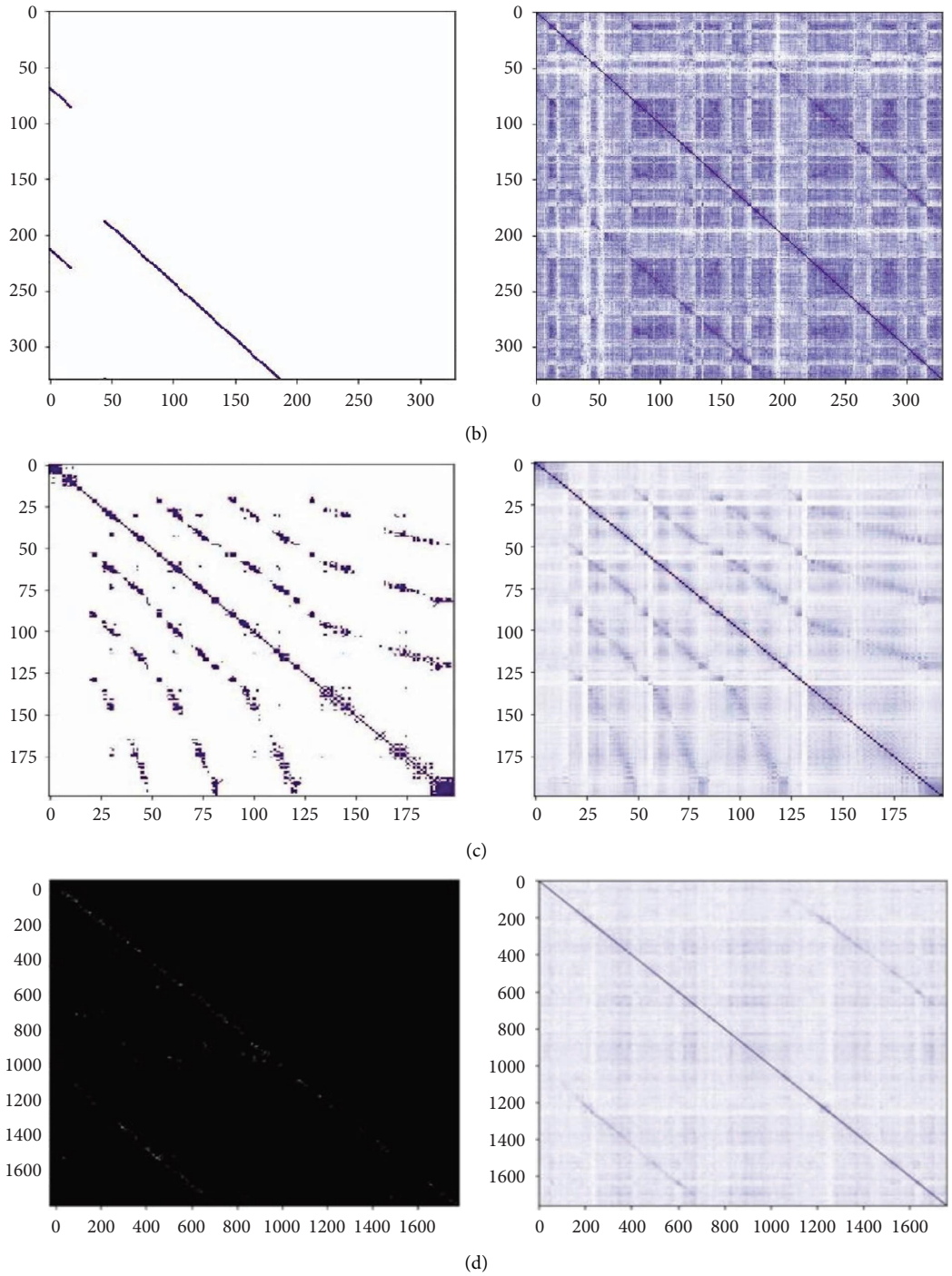


Figure 3: Ground truth (left) and confusion (right) matrices are depicted for the datasets New College, Lip6indoor, Newer College, and St. Lucia in (a–d), respectively.

necessary for the test dataset. Unfortunately, we do not have a ground truth matrix for the Newer College (short experiment) dataset (as shown in Table 4); therefore, we used the HGCN-BoW algorithm to create a similar matrix and regard it as our ground truth.

Table 5 shows the result of applying two state-of-the-art different methods described in [29]. According to Table 3, HGCN-FABMAP outperforms the long-range algorithm described in reference [29] for the St. Lucia dataset; however, the method described in reference is superior to our algorithm.

Table 5: Comparing methods described in [29, 29] and for LCD.

Methods	[29]		[30]	
Dataset	Saint Lucia		New College	
	Precision	Recall	Precision	Precision
	35%	62%	100%	35%

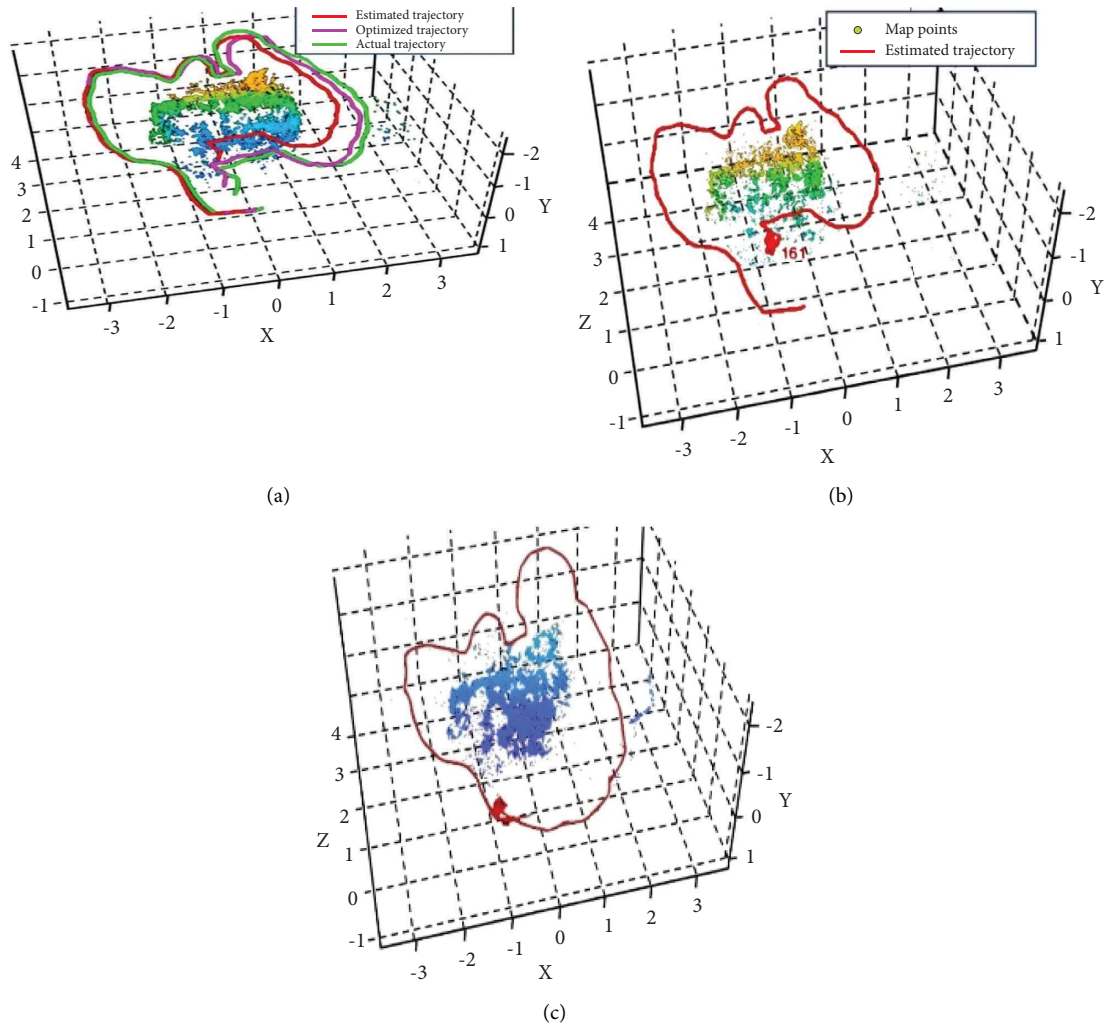


Figure 4: Trajectory of Frieburg-3 dataset: (a) ground truth, (b) ORB-SLAM, and (c) HGCN-ORB before applying pose graph optimization.

Table 6: Comparing new methods against HGCN-ORB-SLAM.

Methods	ORB-SLAM2	ORB-SLAM	HGCN-ORB	IBuild	LSD-SLAM	FOVIS
RMSE	0.010	0.05	0.10	0.30	0.38	0.51

Finally, we compare the ORB-SLAM approach against HGCN-ORB-SLAM. Although we do not outperform the algorithm, the results are competitive. The absolute RMSE for keyframe trajectory (m) in ORB-SLAM is 0.054353, while it is 0.1 in our technique. We modified the MATLAB package provided by Mathworks. Figure 4 depicts a comparison of these two methods.

In Figure 4, we observe a comparison between two algorithms: ORB-SLAM and HGCN-ORB-SLAM. Although our proposed algorithm does not perform better than the ORB-SLAM algorithm, it can be said that these two algorithms have a close competition with each other. The result of executing the algorithm is a trajectory that provides us with the difference in path compared to the ground truth, measured in meters.

Table 6 shows the results of the comparison between HGCN-ORB-SLAM and several up-to-date and state-of-the-art trajectory mapping algorithms. It can be observed that our method is superior to FOVIS, LSD-SLAM, and the method described in references [21, 29, 30].

5. Conclusion and Future Works

In this work, we introduced extensions to current state-of-the-art SLAM algorithms and compared them to FABMAP2, ORB-SLAM, BoW, and several other state-of-the-art algorithms. We showed that clustering in the vector quantization component of the SLAM improves loop closure detection accuracy and recall greatly. Our next goal is to make HGCN-FABMAP work over longer distances. This may not be possible with hyperbolic graph convolutional neural networks, because HGCNs need the whole graph available in RAM at once, and as the number of images and data increases, the task becomes intractable with common computers and servers. We can, however, use semisupervised.

Latent Dirichlet allocation (LDA) is used to first calculate local clusters for each dataset and then merge the resulting clusters to generate global clusters in the second phase.

Appendix

Review of FABMAP1

The following is a brief tutorial on the FABMAP model architecture, adapted from [1, 9].

The FABMAP1 model builds a generative model for a bag-of-words data based on the idea that words that co-occur are typically from the same environment.

This system employs a bag-of-words representation of the received data, which was chosen due to the reduction in learning and inference complexity it provides.

Let $Z_k = \{z_1, ..., z_{|v|}\}$ denote a local scene observation at time k, where z_i is a binary variable indicating the presence or absence of the i^{th} word in the vocabulary and Z^k represents the set of all observations up to time k.

At time k, the algorithm constructs a map from a set of locations $\mathscr{L}^k = \{L_1, ..., L_{n_k}\}$, each of which is associated with an appearance model.

The appearance model of a location is just a set of probabilities of scene elements that exist at that location, i.e., $L_i = \{p(e_i = 1|L_i), ..., p(e_{|v|} = 1|L_i)\}$ where every e_i is a scene element that is generated independently for that location. The elements e_q and the feature z_q are linked using a detector model as follows:

$$D = \begin{cases} p(z_q = 1|e_q = 0), \text{false positive probability}, \\ p(z_q = 1|e_q = 0), \text{false negative probability}. \end{cases} \quad (A.1)$$

The purpose of introducing scene elements is first to create a natural framework for combining data from many sources with various error formulations. Second, it allows us to split the $p(Z \mid L_i)$ distribution into two sections. The first part is a simple model made up of variables called e_q. The second and more complicated part, which is built offline and can be combined with a simple model based on the assumption that conditional dependencies between appearance words are independent of location and how this is achieved, explains the localization and mapping procedure through the use of Bayes filters.

Imagine the robot is in the middle of the path, taking images, and we only have a partial map. When the robot acquires new data, we calculate the likelihood of being in each place, assuming we have all of the observations acquired thus far.

Inference in FABMAP

Calculating probability $p(L_i \mid \mathscr{Z}^k)$ for each place L_i is formulated as follows:

$$P(L_i \mid Z^k) = \frac{p(Z_k \mid L_i, Z^{(k-1)})p(L_i \mid Z^{(k-1)})}{p(Z_k \mid Z^{(k-1)})}, \quad (A.2)$$

in which $p(L_i \mid \mathscr{Z}^{k-1})$ is the prior probability about the location and $p(Z_k \mid L_i, \mathscr{Z}^{k-1})$ is the observation likelihood, and the denominator is the normalization term, taking independence criteria between current and past.

Owing to the high order of the conditional dependency between appearance words in calculating $p(Z_k \mid L_i)$, this term is estimated using the naive Bayes formula:

$$p(Z_k \mid L_i) = \prod_{q=1}^{|v|} P(Z_q \mid L_i), \quad (A.3)$$

$p(z_q \mid L_i)$ can be expanded as follows:

$$p(z_q \mid L_i) = \sum_{s \in \{0,1\}} p(z_q \mid e_q = s)p(e_q = s \mid L_i).[zwj]. \quad (A.4)$$

Furthermore, assuming errors are independent of position in the world $p(z_q \mid e_q, L_i) = p(z_q \mid e_q)$, therefore,

$$p(z_q \mid L_i) = \sum_{s \in \{0,1\}} p(z_q \mid e = L_i)p(e_q = s \mid L_i). \quad (A.5)$$

For the sake of tractability, we assume $p(e_q = S_{s_q} \mid z_{p_q}, L_i) = p(e_q = s_{e_q} \mid L_i)$ which yields

$$p(Z_q \mid Z_{p_q}, L_i) = \sum_{s_{e_q} \in \{0,1\}} p(Z_q \mid e_q = s_{e_q}, z_{p_q})p(e_q = s_{e_q} \mid L_i). \quad (A.6)$$

To derive the probability that an observation came from somewhere other than the map, we must compute $p(z^k \mid z^{k-1})$, which translates the appearance likelihood into a probability of loop closure. In order to calculate $p(z^k \mid z^{k-1})$, our space must be partitioned into mapped and unmapped parts:

$$p(Z^k \mid Z^{(k-1)}) = \sum_{(m \in L^k)} p(Z^k \mid L_m)p(L_m \mid Z_{(k-1)}) + \sum_{(u \in L^k)} p(\hat{Z}k \mid L\_u)p(L\_u \mid Z\_((k-1))). \quad (A.7)$$

Unfortunately, the second term cannot be evaluated directly because it contains unknown locations; however, this part can be approximated using mean-field approximation:

$$p\left(Z_k \mid L_{avg}\right) = \sum_{u \in \overline{\mathscr{L}^k}} p\left(L_u \mid Z_{k-1}\right), \tag{A.8}$$

where $\sum_{u \in \overline{\mathscr{L}^k}} p(L_u \mid Z_{k-1})$ is the prior probability of being in a new place and $p(Z_k \mid L_{avg})$ is the average place.

Data Availability

All data supporting the findings of this study are available within the article. Raw data and materials used in the analysis are available upon reasonable request to the corresponding author.

Disclosure

We would like to note that a preprint version of this manuscript has previously been published [31]. The preprint can be found at https://arxiv.org/abs/2207.06738 [32].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

We would like to express our gratitude to all those who have contributed to this research and provided valuable feedback throughout the development, specifically Dr. Saeid Jalali and Farhad Jalali who provided the high-quality version of Figure 4.

References

[1] B. Xue, Y. He, F. Jing, Y. Ren, L. Jiao, and Y. Huang, "Robot target recognition using deep federated learning," *International Journal of Intelligent Systems*, vol. 36, pp. 7754–7769, 2021.

[2] M. C. dos Santos, R. H. C. Palácios, M. Mendonça, J. A. Fabri, and W. F. Godoy, "A neural autonomous robotic manipulator with three degrees of freedom," *International Journal of Intelligent Systems*, vol. 37, no. 9, pp. 5597–5616, 2022.

[3] J. Kramer and A. Kandel, "On accurate localization and uncertain sensors," *International Journal of Intelligent Systems*, vol. 27, no. 5, pp. 429–456, 2012.

[4] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 4869–4880, Vancouver, Canada, December 2019.

[5] K. Hajebi, Y. Abbasi-Yadkori, H. Shahbazi, and H. Zhang, "Fast approximate nearest-neighbor search with k-nearest neighbor graph," in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1312–1317, Vancouver, Canada, August 2011.

[6] K. Hajebi and H. Zhang, "An efficient index for visual search in appearance-based SLAM," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 353–358, Hong Kong, China, May 2013.

[7] N. Kejriwal, S. Kumar, and T. Shibata, "High-performance loop closure detection using bag of word pairs," *Robotics and Autonomous Systems*, vol. 77, pp. 55–65, 2016.

[8] S. Khan and D. Wollherr, "Lbuild: Incremental Bag of Binary Words for Appearance-Based Loop Closure Detection," in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA, May 2015.

[9] E. Garcia-Fidalgo and A. Ortiz, "ibow-lcd: an appearance-based loop closure detection approach using incremental bags of binary words," *IEEE Robotics and Automation Letters*, vol. 3, pp. 3051–3057, 2018.

[10] M. Ferrera, "Fully online and versatile visual SLAM for real-time applications," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1399–1406, 2021.

[11] M. Cummins and P. Newman, "Appearance-only SLAM at large scale with FAB-MAP 2.0," *The International Journal of Robotics Research*, vol. 30, pp. 1100–1123, 2011.

[12] R. Mur-Artal, J. Montiel, and J. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[13] G. Klein and D. Murray, "Parallel tracking and mapping for small workspaces," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nara, Japan, November 2007.

[14] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge, UK, 2004.

[15] T. N. Kipf and M. Welling, "Semi-supervised Classification with Graph Convolutional Networks," 2017, https://arxiv.org/abs/1609.02907.

[16] G. Bachmann, G. Bécigneul, and O.-E. Ganea, "Constant Curvature Graph Convolutional Networks," 2020, https://arxiv.org/abs/1911.05076.

[17] I. Chami, *CS224W: Machine Learning with Graphs Lecture 19*, Stanford University, Stanford, CA, USA, 2021.

[18] Y. Zhang, "Lorentzian graph convolutional networks," in *Proceedings of the Web Conference 2021*, pp. 1249–1261, Association for Computing Machinery, New York, NY, USA, 2021.

[19] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021.

[20] P. Gao and H. Zhang, "Long-term loop closure detection through visual-spatial information preserving multi-order graph matching," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 06, pp. 10369–10376, 2020.

[21] J. C. V. Soares and M. A. Meggiolaro, "Keyframe-based RGB-D SLAM for mobile robots with visual odometry in indoor environments using graph optimization," in *Proceedings of the 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*, João Pessoa, Brazil, May 2018.

[22] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[23] A. Angeli, D. Filliat, S. Doncieux, and J. A. Meyer, "Fast and incremental method for loop-closure detection using bags of visual words," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1027–1037, 2008.

[24] Tum Vision Group, "TUM Mono Sequence-11 SLAM Dataset," 2022, https://cvg.cit.tum.de/data/datasets/mono-dataset.

[25] A. Glover, W. Maddern, M. Milford, and G. Wyeth, "FAB-MAP+ RatSLAM: appearance-based SLAM for multiple times of day," in *Proceedings of the The International Conference on Robotics and Automation*, pp. 3507–3512, Anchorage, AL, USA, May 2010.

[26] M. Ramezani, "The newer College dataset: handheld LiDAR, inertial and vision with ground truth," in *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4353–4360, New York, NY, USA, December 2020.

[27] A. J. Glover, W. P. Maddern, M. J. Milford, and G. F. Wyeth, "St lucia," 2012, https://code.google.com/archive/p/openfabmap/downloads/openFABMAPsample.

[28] J. Sturm, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 573–580, Vilamoura-Algarve, Portugal, October 2012.

[29] J. Engel, T. Schops, and D. Cremers, "LSD-SLAM: large-scale direct monocular SLAM," in *Proceedings of Computer Vision – ECCV 2014: 13th European Conference*, pp. 834–849, Springer, Berlin, Germany, 2014.

[30] A. S. Huang, A. Bachrach, P. Henry et al., "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Symposium on Robotics Research (ISRR)*, Springer, Berlin, Germany, 2011.

[31] A. Zarringhalam, S. S. Ghidary, and A. M. Khorasani, "Semi-supervised vector-quantization in visual SLAM using HGCN," 2022, https://arxiv.org/abs/2207.06738.

[32] Q. Zhong and X. Fang, "A BigBiGAN-based loop closure detection algorithm for indoor visual SLAM," *Journal of Electrical and Computer Engineering*, vol. 202110 pages, 2021.