AUT Journal of Mathematics and Computing

AUT J. Math. Comput., 7(2) (2026) 117-135
https://doi.org/10.22060/AJMC.2024.23054.1224

Original Article

Loop closure detection in visual appearance-based SLAM using deep autoencoders

Amir Zarringhalam[a], Ali Mohades[*a], Saeed Shiry Ghidary[a,b]

[a] *Department of Mathematics and Computer Science, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran*
[b] *Staffordshire University, School of Digital, Technologies and Arts, College Rd, Stoke-on-Trent ST4 2DE, United Kingdom*

ABSTRACT: Loop closure detection (LCD) and trajectory generation are critical components of visual simultaneous localization and mapping (vSLAM). In this paper, we aim to solve the LCD and trajectory generation problem in vSLAM using a newly devised vector quantization (VQ) algorithm. The proposed new VQ algorithm is constructed based on a selfsupervised deep convolutional autoencoder (AE). The new VQ step is then incorporated into the two famous SLAM algorithms fast appearance-based mapping (FABMAP) and ORB-SLAM, which we now call AE-FABMAP and AE-ORB-SLAM, respectively. Experiments show that using self-supervised autoencoders in the VQ step is far more efficient in terms of speed and memory consumption with respect to other methods such as graph convolutional neural networks. Furthermore, the newly presented algorithms, AE-ORB-SLAM and AE-FABMAP outperform the standard FABMAP2 and ORB SLAM, and in large-scale SLAM, the new approaches improve the accuracy and recall of the LCD.

Review History:

Received:13 March 2024
Revised:13 July 2024
Accepted:02 August 2024
Available Online:01 May 2026

Keywords:

SLAM
Deep autoencoder
Loop closure detection
Vector quantization

MSC (2020):

68T50; 68T07; 68Q55

1. Introduction

Humans can benefit from autonomous vehicles and robots that operate in inhospitable environments such as underground operations and navigation on the surface of other planets. Current autonomous navigation systems rely heavily on the Global Positioning System (GPS) for localization and navigation, which may not be available in many situations, such as densely populated urban areas where skyscrapers obscure satellite visibility. Also, GPS does not work reliably in indoor environments.

More generally, GPS may fail to provide us an accurate enough localization. In such situations, robots can employ visual SLAM as an alternative means of guidance.

Among all proposed solutions to the SLAM problem, the Extended Kalman Filter (EKF) and Rao-Blackwellized particle filter [13] are the most noted, but each of these methods has its own difficulty when applied to large-scale scenarios, as we will briefly discuss below.

[*] *Corresponding author.*
E-mail addresses: am.zarringhalam@aut.ac.ir (A. Zarringhalam), mohades@aut.ac.ir (A. Mohades), saeed.shiryghidary@staffs.ac.uk, shiry@aut.ac.ir (S. Shiryghidary)

2783-2287/© 2026 The Author(s). Published by Amirkabir University of Technology Press.
This is an open access article under the CC BY-NC 4.0 license (https://creativecommons.org/licenses/by-nc/4.0/)

UNCORRECTED PROOF

1.1. Extended Kalman Filter

Suppose our map consists of landmarks and an autonomous vehicle. In the EKF approach the filter assumes that both landmark and vehicle covariance are approximated by Gaussian distributions. Although the Kalman Filter-based approach is used widely and is one of the prominent SLAM solutions it suffers from the following several performance-limiting issues:

- Computational scaling
- Linearization
- Gaussianity assumption

In [40] and [5], sub-sampling strategies are utilized to alleviate scalability issues (The fact that the computation required to maintain a map grows quadratic in proportion to the number of landmarks). Secondly, Kalman Filter is only applicable to linear processes. Through linearization of all functions about their current estimations, EKF expands the approach when the functions are highly non-linear. Although methods developed by [32] and [3] put remedies for this problem, still linearization problem remains an essential and inevitable problem of the Extended Kalman Filter. Finally, the worst drawback of EKF-based is the presumption that both vehicle and landmark covariances are well represented by Gaussian distribution where this is not always the case, especially when the probability density function is multi-modal.

1.2. Rao-Blackwellized Particle Filter Method

This particle filter SLAM presents a solution based on the Rao-Blackwellized particle filter. This solution takes advantage of the presumption that given the vehicle's trajectory, the approximation of the landmark positions is conditionally independent. This solution is named FastSLAM. In contrast to the EKF technique, FastSLAM properly depicts any stochastic patterns across the vehicle's position. This results from nonlinear vehicle kinematics where position uncertainty is multi-modal. Furthermore, it does not suffer from linearization issues. The only difficulty with FastSLAM is the loss of information about trajectory history at large-scale navigation.

In large-scale SLAM, localization, and positioning tend to drift for autonomous vehicles that plan to drive hundreds of kilometers in varying environmental conditions. Additionally, maps do not always remain viable under different driving conditions, and positioning tends to deviate from the true trajectory making it difficult to ensure correct localization without prior knowledge of the environment. To address this challenge, researchers have proposed creating sub-maps and multiple maps, ([42], [16], and [50]). However, these approaches rely on proper data association assumptions and require consistent improvement to build accurate large-scale maps.

Current challenges and open areas in vSLAM are motion planning, obstacle avoidance, and loop closure detection (LCD) for mobile robots. As discussed in [34] motion planning is still among the most challenging areas. Its goal is to interpret the high-level languages into a set of primary low-level movements. Authors in [34] have tackled this issue using adaptive Adaptive Neuro Fuzzy Inference System, (ANFIS).

In this study, we focus on the most essential part of visual SLAM called loop closure detection (LCD), which determines if a robot has returned to a previously visited region after a period of exploration [22]. Loop closure detection (LCD) is a crucial section of the SLAM algorithms in generating accurate global maps and minimizing cumulative pose errors in robot localization. It can also correct the robot's position if necessary. The critical element in solving the LCD problem is the vector quantization (VQ) algorithm. Previous methods for addressing large-scale LCD problems have typically used unsupervised VQ algorithms, and there has been no known deep learning-based approach for constructing VQ algorithms until now. The significant advantage of the self-supervised methods proposed in this study is their reduced computational time and space complexity. Our experiments show that the space complexity of our approach is approximately 1/100 that of semi-supervised methods, and the execution time is around 1/10 of hyperbolic graph convolutional neural network-based methods. We have employed FABMAP2 and ORB-SLAM as our primary platforms for creating AE-FABMAP and AE-ORB-SLAM. Our findings indicate that the proposed AE-FABMAP algorithm surpasses the performance of the standard FABMAP2 algorithm. The reason for selecting the FABMAP2 platform is due to its utilization of a strong Bayesian network, BoW-based approach, and ability to achieve loop closures in extensive trajectories with recall rates high enough for accurate mapping with high precision, as demonstrated in previous research by [24]. The paper is structured as follows: Section 2 reviews previous works, Section 3 will focus on the main algorithms and innovation ofthe proposed research. The experiment are presented in Section 4, finally Section 5 will cover the discussion, conclusion, and future directions.

UNCORRECTED PROOF

2. Related Works

Numerous studies have attempted to tackle loop closure detection in large-scale trajectories through the BoW approach. For instance, [24] and [23] proposed a stochastic graph nearest neighbor (SGNNS) search algorithm, which is currently considered the most efficient in terms of speed. This method involves constructing a graph from features extracted from a sequence of images. SGNNS begins at a random node and applies a threshold to the nearest neighbor of the node, using KNN search to cluster the features. The main benefit of SGNNS over linear search and K-means is its superior speed. However, the nearest neighbor distance constraint imposed by this method is limiting. [27] proposed a bag of words pair (BoWP) based loop closure detection method that resolves the limitations of the simple BoW method like perceptual aliasing. This is achieved by incorporating the spatial occurrence of words in the vector quantization procedure. They use a Bayesian probabilistic scheme for loop closure detection. [20] proposed an incremental BoW LCD (iBoWLCD) method that uses binary features to generate an online BoW without deleting any visual words learned during training. [19] used a hierarchical BoW scheme to detect revisited places and constructed a tree-structured hierarchical BoW using a set of binary features discretized to k cluster centroids. The set of clusters constructed in this stage forms the first level of the tree. This procedure is repeated on every cluster to make the subsequent levels of the tree. Khan et. al., [28], proposed an online, appearance-based loop closure detection method called Incremental Bag of Binary Words for Appearance-based Loop Detection (IBuILD). This method builds incremental binary vocabularies and tracks features between two consecutive images to detect loop closures, without requiring GPS estimation or odometry data. A simple likelihood function and provisional consistency constraints are used to generate suitable loop closure detection candidates and filter non-homogeneous closures out.

[29], proposed a method for pose graph optimization that enhances loop closure detection performance. This method uses two novel metrics, expected information gain, and visual word saliency, which uses global and local saliency to measure the scarcity of an image in a dataset and the richness of candidate images for loop closure detection, respectively. The later measure is defined using a bag of visual words histogram entropy for key-frames.

The camera pose detection method for an unknown scene is presented in Parallel Tracking And Mapping (PTAM), and its alternative version (s-PTAM) is presented in [30] and [41] which are the foundation for ORB-SLAM. They introduce a novel parallel approach for tracking and mapping in a small-scale augmented reality setting. This approach involves two threads working simultaneously, one tracking the cluttered movements and the other building a 3D map using previously observed frames' extracted features. Parallel processing allows for costly batch optimization operations to be carried out in real-time, resulting in maps with thousands of accurately tracked landmarks per frame. This makes costly batch optimization operations feasible in real time, which results in maps with thousands of landmarks that can be tracked per frame with sufficient accuracy.

DTAM is the first direct algorithm that produces dense map by means of dense mapping and dense tracking. The dense tracking DTAM estimates the vSLAM motion equations parameters by means of matching and aligning an image projected on the camera and current frame [39] and [18].

LSD-SLAM is a direct method that constructs a semi-dense maps. This method is comprised from 3 main modules. Tracking, depth-map estimation, and the map global optimization in the first stage, it estimates the sensor pose through minimizing the photometric error.

LSD-SLAM in depth estimation stage selects a set of keyframes and if it adds the previously selected keyframe to the algorithm, it should initialize the depth estimation, [14], [4], and [1].

None of the algorithms, MonoSLAM, PTAM, SVO, DTAM, and DSO detect loop closures. Among others, Ultrametric–FABMAP supports this feature. However, this method is not practical for real-size datasets. Furthermore, methods HGCN-FABMAP and HGCN-ORB support this feature, but they have a high time and space complexity. This is due to the fact that these two algorithms need the whole graph data to be available in RAM at once. So among the remaining algorithms ORB, AE-ORB, and AE-FABMAP support Loop Closure Detection (LCD). However, AE-ORB, in comparison with other methods is more accurate while mapping the trajectory. For the AE-FABMAP case, as the experiments show, it has a higher accuracy and recall for LCD with respect to other algorithms such as FABMAP and HGC-FABMAP.

For map density, we have focused on long-range SLAM and LCD algorithms. AE-ORB is among those that produce sparse maps suitable for such applications. AE-FABMAP and HGCN-FABMAP can be integrated with a SLAM system, but HGCN-FABMAP requires a huge time and space complexity.

For Global Optimization (GO), that are a set of processes and techniques used to refine a trajectory consistency, AE-ORB is among those that support GO.

Finally, AE-ORB is a lightweight algorithm suitable for embedded systems.

UNCORRECTED PROOF

Method	Type	Map Density	Global Optimum (GO)	LCD	Embedded Implementation	Availability
Mono-SLAM	Feature-based	Sparse	No	No	[48], [49]	[11]
PTAM	Feature-based	Sparse	Yes	No	[31]	[10]
DTAM	Direct	Dense	No	No	[39]	[18]
SVO	Hybrid	Sparse	No	No	[12]	[17]
LSD	Direct	Semi-Dense	Yes	Yes	[14], [4]	[1]
ORB-SLAM	Feature-based	Sparse	Yes	Yes	[2]	[35], [44]
CNN-SLAM	Direct	Semi-Dense	Yes	Yes	[51]	[45]
DSO	Direct	Sparse	No	No	[9]	
Ultrametric-FABMAP	Loop Closure Detection	Semi-Dense/ Sparse	Enables GO	Yes		[55]
HGCN-ORB	Feature-based	Sparse	Yes	Yes		[56]
HGCN-FABMAP	Loop Closure Detection	Semi-Dense/ Sparse	Enables GO	Yes		[56]
AE-ORB	Feature-based	Sparse	Yes	Yes		[53]
AE-FABMAP	Loop Closure Detection	Semi-Dense/ Sparse	Enables GO	Yes		[52]

3. Methods

In this section, we provide a brief overview of the fundamental techniques and our suggested alterations employed in the paper. We begin by giving a brief explanation of deep convolutional autoencoders (AE). Afterward, we provide a concise review of the Chow-Liu technique and the FABMAP2 approach and present our innovative algorithm, AE-FABMAP, which employs an AE for vector quantization. Additionally, we touch on ORB-SLAM and our modified version, AE-ORB-SLAM. Finally, we describe our integration of an autoencoder into BoW SLAM.

3.1. Deep Convolutional Autoencoder (AE)

Convolutional Autoencoder (CAE) is a model based on the encoder-decoder paradigm: The encoder transforms the input into a lower dimensional space, and the decoder attempts to reconstruct the original input from the low dimensional representation. This is achieved by minimizing the cross entropy cost function [47]:

$$E = \frac{1}{N}\sum_{n=1}^{N}(y\log(\hat{y_n}) + (1-y_n)\log(1-\hat{y_n})) \tag{1}$$

CAE uses convolution/deconvolution layers for the encoding/decoding part. These layers are followed by an activation function as follows:

$$h^k = f(\sum_{l\in L} x^l \circledast w^k + b^k) \tag{2}$$

In Formula (2), k is the latent representation of the k^{th} feature map of the current layer, f is a non-linear activation function, the symbol $\circledast$ denotes the 1D convolution operation, w^k and b^k are the weights (filters) and bias of the $k-$th feature map of the current layer, respectively. The CAE layers are arranged as follows:

Encoder:

- First layer consists of $32-3\times 1$ filters followed by a max-pooling layer.
- The second layer will have $64-3\times 1$ filters followed by a down-sampling layer.
- The final layer is made up $128-3\times 1$ filters.

Decoder:

- The first layer incorporates $128-3\times 1$ filters, which are then followed by an up-sampling layer.
- The second layer consists of $64-3\times 1$ filters, followed by another up-sampling layer.
- The final layer is composed of a single $1-3\times 1$ filter.

UNCORRECTED PROOF

3.2. Self-supervised Vector Quantization(SVQ)

In the context of image compression and reconstruction, CAE is utilized to create a compressed low-dimensional representation of the input image. This representation is then quantized into discrete values using Vector Quantization (VQ), which involves mapping the input vectors to a finite set of codewords or cluster centroids denoted by i. These centroids belong to a countable set represented by $i \in \{1, .., |\mathcal{C}|\}$, where $|\mathcal{C}|$ is the number of clusters obtained from a convolutional deep Autoencoder applied to the extracted SURF features of the images. To provide a more detailed explanation, let D be the set of descriptors for a particular image in the database:

$$\hat{Q(D)}_i = \{j : \|D[i,] - C[j,]\|_2 \leqslant \|D[i,] - C[k,]\|_2 \forall k \in \{1, .., |\mathcal{C}|\}\} \tag{3}$$

is the quantized representation of features in that particular image.

In what follows a brief description of Chow-Liu trees, which is a fundamental unit of FABMAP2 and AE-FABMAP, is presented.

3.3. Chow-Liu Trees

Working with a large number of random variables in a discrete distribution, e.g. more than 100,000 variables becomes intractable. We need to keep the whole information of the distribution in the following way: only keep the ones that predict others with large weights (edges) in terms of mutual information between the nodes (random variables). Chow-Liu tree gives an approximation for this distribution $P(z)$ by closest Bayesian network tree structure, $Q(z_{opt})$. The aim is to retain the maximum weight spanning tree by eliminating the not-very-significant edge weight in terms of the mutual information. We assume that the edges are directed so that each node has precisely one parent. To find a fixed tree, we need to solve the following optimization problem:

$$\text{Max} \log l(\theta, T) = C + \sum_{(i,j)\in E(T)} \sum_{x_i, x_j} N_{x_i,x_j} \log \frac{\mathrm{N}_{x_i,x_j}}{\mathrm{N}_{x_i}\mathrm{N}_{x_j}} \tag{4}$$

Where C is independent of the tree structure and the second term denotes empirical mutual information, $I(x_i, x_j)$. To maximize log-likelihood, simply select a tree that maximizes

$$\underset{T}{\text{Max}} \sum_{i,j} I(x_i, x_j) \tag{5}$$

This is the same as determining the maximum weight spanning tree in a complete graph, a greedy method will pick the largest edge each time while avoiding generating a cycle when added to the already picked edges. To do so, we must compute mutual information for each edge using an empirical probability distribution.

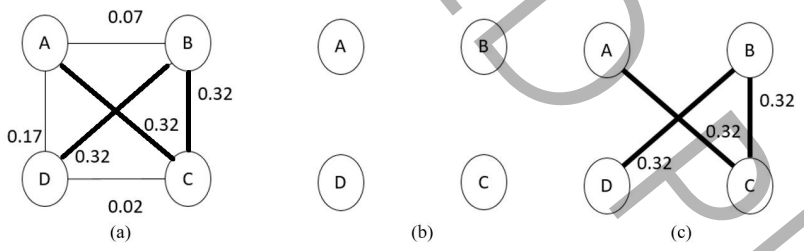

Figure 1: (a) Discrete distribution tree of the variable. (b) Bayesian estimation. (c) Chow-Liu's approximation

In the figure 1 part (a) the whole data (distribution) is present while in part (b) the naïve Bayes assumption (independence of variables) is held. Part (c) shows the Chow-Liu assumption that holds only significant edge weights. For more information about the Chow-Liu tree, see section 3.1 in [7].

3.4. FAB-MAP2

The following is a brief overview of the FABMAP2 algorithm. For a more detailed explanation, refer to [38] and [7]. FABMAP2 is a generative model that assumes that the words that co-occur in an image originate from the same place, and constructs a Bag of Words (BoW) representation of images based on this assumption. This enables FABMAP2 to detect places having too many features in common. In this method, the observation Z of an image at time k is reduced to a binary vector $Z_k = \{z_1, ..., z_{|\mathcal{C}|}\}$, where z_i indicates the presence or absence of the word i in the vocabulary. The vocabulary is obtained by clustering the extracted SURF features of the images using

UNCORRECTED PROOF

Kmeans. The overall observations up to time k are denoted by Z^k. The map of the environment is constructed from locations $\mathcal{L}^k = \{L_1, ..., L_{n_k}\}$. Each of these locations is mapped to an appearance model, for example, $L_i = \{p(e_i = 1|L_i), ..., p(e_{|\mathcal{C}|} = 1|L_i)\}$. Assuming the robot is currently located halfway through its path and has generated a partial map of the surrounding environment, as the robot gathers new observations, the likelihood of its presence at each location (represented as $p(L_i|Z^k)$) is determined based on the observations collected up to that point in time:

$$P(L_i|\mathcal{Z}^k) = \frac{p(Z_k|L_i, \mathcal{Z}^{k-1})p(L_i|\mathcal{Z}^{k-1})}{p(Z_k|\mathcal{Z}^{k-1})} \tag{6}$$

Formula (6) entails that, $p(L_i|\mathcal{Z}^{k-1}))$ represents the prior probability of the robot's location, while $p(Z_k|L_i, \mathcal{Z}^{k-1})$ corresponds to the observation likelihood.
To simplify the evaluation of $p(Z_k|L_i, \mathcal{Z}^{k-1})$, it is assumed that observations in the current time and the previous time are independent. Therefore, applying the naive Bayes formula yields:

$$p(Z_k|L_i) \approx p(z_r|L_i)\prod_{q=1}^{|\mathcal{C}|} p(z_q|z_{P_q}, L_i) \tag{7}$$

To determine the probabilities of unobserved areas on the map, it is necessary to compute $p(z^k|z^{k-1})$ and partition the map space into two segments: mapped and unmapped regions.

$$p(Z^k|Z^{k-1}) = \sum_{m\in\mathcal{L}^k} p(Z^k|L_m)p(L_m|Z_{k-1}) + \sum_{u\in\overline{\mathcal{L}}^k} p(Z^k|L_u)p(L_u|Z_{k-1}) \tag{8}$$

In Formula (8), the second term cannot be calculated directly due to the presence of unexplored areas. However, the mean-field approximation can be employed to estimate this term, as follows: $p(Z_k|L_{avg}) = \sum_{u\in\overline{\mathcal{L}^k}} p(L_u|Z_{k-1})$, where $\sum_{u\in\overline{\mathcal{L}^k}} p(L_u|Z_{k-1})$ is the posterior probability of being in a new place.

3.5. AE-FABMAP

To quantize the image features in AE-FABMAP, we substituted the conventional unsupervised method, Kmeans, with an autoencoder. Essentially, we extract the features from the image database and input them into a 19-layer convolutional autoencoder to generate labels, which are then employed to produce cluster centroids and the BoW representation of the images. We subsequently fuse the BoW representation and cluster centroids obtained in the previous stage into the FABMAP2 method to find the similarity between images in form of a confusion matrix. In the AE-FAMBAP algorithm, the main LCD criteria is as follows: If the current observation matches a location with a probability higher than a user-specified threshold (e.g., p > 0.999), we associate the observation with that location, [7]. A detailed description of AE-FABMAP is given in algorithm .

The algorithm starts by initializing the descriptor of the test images, This step involves calculating the default likelihood once a location is added to the map for the first time. This is accomplished by initializing the location's default likelihood with d_1, wherein a null observation with $z_q = 0$. It continues by iterating through test images and creates an inverted map index to keep track of visual words occurring in the images, If a visual word has been seen in the query image, the algorithm updates the likelihood of other images in the database containing that visual word. The update is also dependent on the parent node of this feature, If the parent feature is present, it adds a specified value to $d4[q]$, otherwise, it assigns a different value ($d3[q]$). The output of algorithm , provides us with the degree of correspondence between the query image and the images in the database, along with the quantification of the likelihood of each correspondence. The computed probabilities are based on the presence or absence of the specific features and their hierarchical connections (parent and child). For each image, q is the index of the word in the vocabulary and ranges from 0 to the $|\mathcal{C}|$. Filling the confusion matrix can easily be achieved using the filled matches vector. Log-likelihoods are calculated using d_3 and d_4 based on the content of the current observation, and d_2 is used to compute log-likelihoods for unobserved words that are children of observed words in the Chow-Liu tree.

Formulas used in algorithm

We have tried to explain the formulas used in our method. More information can be obtained from Appendix section.

$$d1^* = \frac{(1 - \text{CLtree}(1, q)) \times 0.61 \times (1 - \text{CLtree}(3, q))}{\text{CLtree}(1, q) \times 0.39 \times (1 - \text{CLtree}(3, q)) + (1 - \text{CLtree}(1, q)) \times 0.61 \times (1 - \text{CLtree}(3, q))} \tag{9}$$

UNCORRECTED PROOF

Algorithm AE-FABMAP

Input: Training and Testing Images
Output: Confusion Matrix

```
Vector <Imatch> & matches
Vector test_Image_Descriptor
for i = 0 ; # test _images ; i++ do
    Vector <Imatch> QueryMatch
    Vector <Double> Default
    map <int, Vector <int> > inverted_map;
    for k = 0 ; # test _images ; k++ do
        Default.push_back(0);
        for q = 0 ; q < CLtree(0,q) ; q++ do
            if test_images(k)(0,q) > 0 then
                Default.[end] += d1[q]
                inverted_map[q].append(Default.size())
            end if
        end for
        Vector::iterator Lwidth, child;
        Vector<Double> liklihoods = Default;
        for q1 = 0 ; q1 < CLtree.cols() ; q1++ do
            if queryImgDescriptor(0,q1) > 0 then
                for Lwidth=inverted_map(q1).begin(); Lwidth!=inverted_map(q1).end(); Lwidth++ do
                    if queryImgDescriptor(0,P_q(q1)) > 0 then
                        liklihoods[*Lwidth]+= d_4[q1]
                    else
                        liklihoods[*Lwidth]+= d_3[q1]
                    end if
                    for child=children(q1).begin(); child!=children(q1).end(); child++ do
                        if queryImgDescriptor(0,*child) == 0 then
                            for Lwidth=inverted_map(*child).begin();
                                Lwidth!=inverted_map(*child).end(); Lwidth++ do
                                liklihoods[*Lwidth]+= d_2[*child];
                            end for
                        end if
                    end for
                end for
            end if
        end for
    end for
    for int k=0 ; liklihoods.size() ; k++ do
        QueryMatch.push_back(Imatch(0,k,liklihoods(k),0));
    end for
    for int j=1 ; QueryMatch.size() ; j++ do
        QueryMatch[j].QueryIndx = i;
    end for
    matches.insert(matches.end(), QueryMatch.begin(), QueryMatch.end());
end for
```

$$d_1 = \log(d1^*) \tag{10}$$

Starting from d_1 the denominator of formula $d_1[q]$ is a normalization term, in the paper context. It guarantees that d_1 is a valid probability or weighting factor, which equals 1 when the factor is integrated overall potential condition. The numerator of $d_1[q]$ represents a specific scenario when CLtree$(1, q)$ is false where it is the marginal probability of observing the visual word q. Furthermore, CLtree$(3, q)$ represents the conditional probability $P(z_q|z_{p_q} = false)$ and it is the probability of observing the visual word given that its parent word in the Chow-Liu tree is not observed and in the case of d_1 it is false. The equation for d_2 provides a quantification of the logarithmic likelihood of the

UNCORRECTED PROOF

feature q being absent, assuming that its parent feature is likewise absent. This measure is normalized and modified by the conditional probabilities and constants in the Chow-Liu tree model. Here we break down each term of the d_2 formula:

$$d2_{num} = \frac{\text{CLtree}(1,q) \times 0.61 \times (1 - \text{CLtree}(2,q))}{(1 - \text{CLtree}(1,q)) \times 0.39 \times \text{CLtree}(2,q)} \tag{11}$$

$$d_2 = \log(\frac{d2_{num}}{1 - d2_{den}}) - d_q \tag{12}$$

$$A = [(1 - \text{CLtree}(1,q)) \times 0.61 \times (1 - \text{CLtree}(2,q)) + \text{CLtree}(1,q) \times 0.39 \times \text{CLtree}(2,q)] \tag{13}$$

$$d2_{den} = \frac{\text{CLtree}(1,q) \times 0.61 \times \text{CLtree}(2,q)^2 \times 0.39}{A \times (1 - 0.39 \times \text{CLtree}(1,q))} \tag{14}$$

0.61 is an invariant coefficient that might represent a certain probability or significance in the model. The $d2_{num}$'s denominator, $(1 - \text{CLtree}(1,q)) \times 0.39 \times \text{CLtree}(2,q)$, is the likelihood of observing the visual word q provided that its parent feature is not observed. Totally the numerator integrates these probabilities to depict a precise likelihood scenario involving the visual word q and its parent feature. The denominator's numerator, $\text{CLtree}(1,q) \times 0.61 \times \text{CLtree}(2,q)^2 \times 0.39$ refers to a situation in which the marginal probability of q and its conditional probability given the parent feature are both substantially weighted. The weights 0.61 and 0.39 adjust the probabilities, and the squaring of $\text{CLtree}(2,q)$ indicates a reinforcement or interaction effect, implying that the conditional probability of observing q, given its parent feature is true, plays a significant role in this part of the model. The term in the bracket in the denominator essentially considers various probability scenarios, and it provides a comprehensive measure, which incorporates both cases when q is present and when it is absent adjusted by weights. To provide more details, we will consider two different scenarios:

- (Absent q)
  1. $1 - \text{CLtree}(1,q)$: Probability q is absent.
  2. $1 - \text{CLtree}(2,q)$: Probability q is absent given its parent is present.
  3. 0.61 : weighting factor.
- (Present q)
  1. $\text{CLtree}(1,q)$: Probability q is present
  2. $\text{CLtree}(2,q)$: Probability q is present given its parent is present
  3. 0.39 : weighting factor.

By combining these two scenarios, the expression results in a weighted average probability accounting for both the presence and absence of the visual word q, depending on its parent feature.

The details of calculating d_3 are as follows: The numerator takes the chance of q being present into account when its parent is false, in addition to a weighting factor being used. The denominator considers the probability of q being present overall but not when its parent is missing.

$$d3_{den} = \frac{0.61 \times \text{CLtree}(1,q) \times (1 - \text{CLtree}(1,q)) \times 0.39 \times \text{CLtree}(3,q)}{(1 - \text{CLtree}(1,q)) \times \text{CLtree}(1,q) \times 0.61 \times (1 - \text{CLtree}(3,q))} \tag{15}$$

$$d3_{num} = \frac{(1 - \text{CLtree}(1,q)) \times 0.39 \times \text{CLtree}(3,q)}{(1 - \text{CLtree}(1,q)) \times 0.39 \times \text{CLtree}(3,q) + \text{CLtree}(1,q) \times 0.61 \times (1 - \text{CLtree}(3,q))} \tag{16}$$

$$d_3 = \log(\frac{d3_{num}}{d3_{den}}) - d_q \tag{17}$$

Intuitively $d3[q]$ Represents the log-probability of the feature q being present given that its parent feature is absent.

UNCORRECTED PROOF

For d4, the numerator combines the marginal probability of $q's$ presence with a weighting factor, indicating the impact when the parent feature is present. The ratio represents the modified likelihood of q being present given the influence of its parent feature, with probabilities properly normalized to guarantee consistency.

$$d4^* = \frac{\text{CLtree}(1,q) \times 0.61}{1 - 0.39 \times \text{CLtree}(1,q)} \tag{18}$$

$$d4 = \log(d4^*) - d_q \tag{19}$$

Conceptually the $d4[q]$ represents the log-probability of the feature q being present given that its parent feature is also present.

3.6. AE-ORB-SLAM

The process of AE-ORB SLAM is illustrated in figure 2, which follows a similar approach to ORB-SLAM proposed by [35] However, the key distinction is that while ORB-SLAM uses an unsupervised Kmeans clustering algorithm for loop closure detection, AE-ORB-SLAM replaces this component with a self-supervised deep convolutional autoencoder.

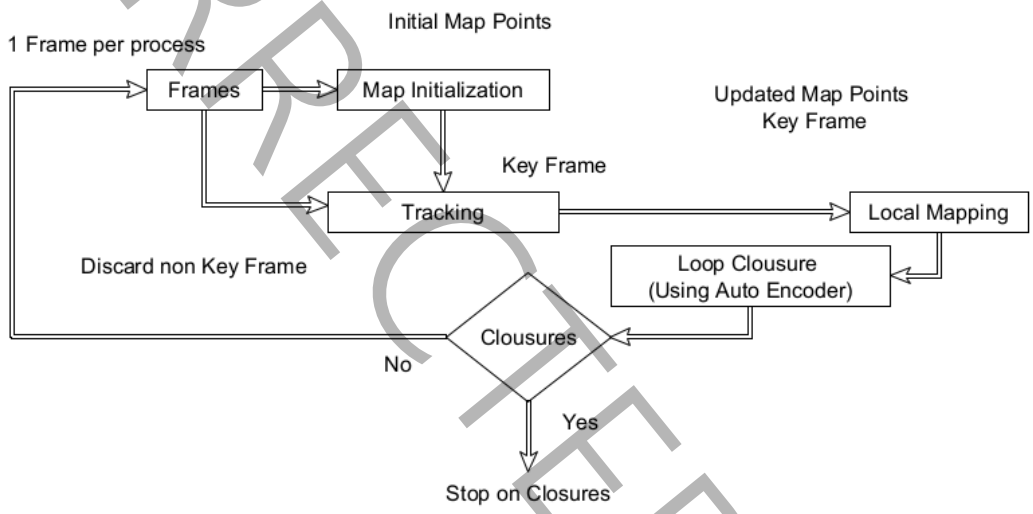

Figure 2: Pipeline of Deep Convolutional AE-ORB SLAM

AE-ORB-SLAM begins with initializing the points in the map in 3D space, followed by triangulation of the ORB features to obtain 3D map points and relative camera position.

During tracking, matching of ORB features is performed, and the estimated camera pose is continuously refined from the first to the last frame. In the local mapping step, a point corresponding to a detected keyframe image is placed in the 3D map using bundle adjustment. Finally, for loop closure detection, a BoW representation of images is constructed offline using a deep convolutional autoencoder.

3.7. AE-BoW

In the traditional bag of words (BoW) method, the representation of images is obtained using cluster centroids generated by applying Kmeans on standard image features. Loop closure detection is then performed by computing cosine similarity between each image and all other images. However, in AE-BoW, we replace Kmeans with a deep convolutional autoencoder to obtain the cluster centroids. This replacement greatly enhances the performance of loop closure detection.

4. Experimental Results

4.1. Metrics

In this article, the evaluation metrics employed are the same as those used in [57], namely the recall and accuracy of loop closure detection. These metrics are defined in formulas (20) and (21).

UNCORRECTED PROOF

In this article we have used the metrics used in [57]. It uses recall and accuracy of loop closure detection, They are defined as follows:

$$Recall = \frac{\sum_i \sum_j ((\text{Confusion Matrix}[i][j] > \text{threshold}) \wedge \text{ground truth}[i][j] == 1)}{\sum_i \sum_j \text{ground truth}[i][j] == 1} \tag{20}$$

$$Accuracy = \frac{\sum_i \sum_j ((\text{Confusion Matrix}[i][j] > \text{threshold}) \wedge \text{ground truth}[i][j] == 1)}{\sum_i \sum_j (\text{Confusion Matrix}[i][j] == 1 > \text{threshold})} \tag{21}$$

Each element (i^{th}, j^{th}) in the ground truth matrix, as stated in equations (20) and (21), represents whether images i and j were taken from the same location. A value of 1 indicates that the images are taken from the same location and 0 otherwise. The true positive value, the numerator in formulas (20) and (21), represents the number of elements in the confusion matrix that are greater than a specified threshold, and whose corresponding elements in the ground truth matrix are also 1.

4.2. Datasets

The methods presented in this paper are tested on several indoor and outdoor datasets. The four methods BoW-SLAM, AE-Bow, FABMAP2, and AE-FABMAP are applied on Lip6indoor dataset, TUM sequence 11, Stlucia (train/test), sequence 6 and 7 from KITTI dataset, [21]. For ORB-SLAM the experiment is performed on TUM Freiburg3 long office household. In the following section, we briefly review the datasets used in this article. AE-FABMAP implementation is an extension to the C++ package provided by [22].

4.2.1. Datasets used in AE-FABMAP, AE-BoW, and AE-ORB-SLAM

Table 1 provides useful information about the datasets that we have used in the experiment section. In addition figure 3 presents sample images from the datasets provided in the Table 1.

4.3. Results

Table 2 presents a comparison between AE-FABMAP, FABMAP2, and HGCN-FABMAP [56], applied to the training datasets Saint Lucia (training), TUM sequence 11, Kitti sequence 6, New College and testing datasets Saint Lucia (testing), lip6indoor, Kitti sequence 7 and Newer College. The evaluation of these methods is based on their accuracy and recall performance on the testing datasets.

Additionally, Table 3 presents a comparison between AE-BoW, BoW, and HGCN-BoW, using the same datasets, and following a similar experimental setup. Furthermore, we compare our introduced method AE-ORB-SLAM to the ORB-SLAM and HGCN-ORB-SLAM methods adopted from [56], [43], [15]. For this comparison, we select training and testing datasets that have similar content without any repeated path.

Despite the competitive results of ORB-SLAM and HGCN-ORB-SLAM, our proposed method AE-ORB-SLAM outperforms both algorithms, as indicated by the lower absolute RMSE for keyframe trajectory (m) of 0.0473, compared to 0.054353 and 0.1 for ORB-SLAM and HGCN-ORB-SLAM, respectively. To implement deep AE-ORB-SLAM, we have employed modifications to the MATLAB ORB-SLAM package.[1]

Table 4 presents a comparison between the proposed AE-ORB-SLAM method and other state of the art methods.

The trajectory obtained from applying algorithms ORB-SLAM and AE-ORB-SLAM are shown in parts (a) and (c) of figure 4 respectively. In this figure, ground truth is plotted in section (b). Furthermore the HGCN-ORB-SLAM trajectory is plotted in part (d).

To visually inspect loop closures, we have plotted the results of AE-BoW applied on the datasets Lip6indoor, New-College, Newer College, and Saint-Lucia in figure 5. The confusion matrices in the figure are consistent with the ground truths of the datasets, and the horizontal and vertical columns represent images, while each datapoint in parts (a), (c), (e), and (g) represents the cosine similarity between the BoW representation of the corresponding images. Brighter data points indicate higher similarity between images in terms of BoW representation.

Table 5 represents the number of existing and detected loops for datasets Newer College, Saint-Lucia, Lip6-indoor, and KITTI 6th sequence.

[1] https://www.mathworks.com/help/vision/ug/monocular-visual-simultaneous-localization-and-mapping.html

UNCORRECTED PROOF

UNCORRECTED PROOF

Table 1: Datasets used in the experiments part along with their description

Dataset Name #Images	Used as Training	Used as Testing	Long Range	Description
Lip6Indoor 387 (Testing) Trained on TUM-seq 11 1500		•		Images are captured from narrow corridors of a lab environment With a resolution of 25×192. The dataset involves multiple loops.
TUM-seq 11 1500	•			This dataset consists of images taken in a lab under various lighting conditions (several rooms are included in the path).
New-College 1073 (Testing) Training on StLucia training sequence	•	•	•	This dataset has been gathered from the Oxford University campus, which includes complex repetitive structures. This is a stereo dataset with a resolution of 640×480, and we use the left sequence.
Newer-College 200 (For Testing) Trained on New-College		•		This dataset is adopted from a video, which we converted into 200 images sequence. The dataset can be considered as a subset of the New-College dataset; it contains 3 loops and has been collected at night. The camera is experiencing cluttered movements.
StLucia suburbs Train Test 540 1000	•	•	•	This dataset has two sets of training and testing movies, compromising roughly 540 and 1000 images respectively. It has been collected from Saint Lucia suburbs, with similar environments in both sets
Freiburg3 2500		•		This indoor dataset contains images captured from the area around a table, and it contains one loop. This dataset Is specifically used in AE-ORB-SLAM.
KITTI seq 6	•		•	The outdoor gray images have been captured using a cameramounted car on a large-scale path.
KITTI seq 7 (For testing) Training on KITTI seq 6		•	•	Same as KITTI seq 6

Table 2: Comparing AE-FABMAP, FABMAP2, and HGCN-FABMAP methods for LCD.

Train Dataset	Test Dataset	Our method (AE-FABMAP)		FABMAP2 [22]		HGCN-FABMAP [56]	
		%Acc	%Rec	%Acc	%Rec	%Acc	%Rec
Stlucia 311417	Stlucia 111417	%74	%79.12	%59.55	%81.5	%83.8	%69.5
TUM seq11 3924499	Lip6Indoor 27189	%60.8	%72.2	%52.59	%81.2	%57.4	%81.6
		%59.55	%81.5				
Kitti seq 6 227904	Kitti seq 7 235510	%54.8	%90.8	%51.1	%53	%51.3	%81.5
		%60	%81.5				
		%53.9	%72.2.8				
New College 160500	Newer College 194000	%64.8	%94.3	%60.1	%89	%64.8	%86.6
		%80.2	%84.7	%74.4	%71.5		

5. Conclusion

In this paper, we attempted to introduce extensions to the current state-of-the-art long-range SLAM methods and compared them with FABMAP2, ORB-SLAM, and BoW. Our findings demonstrate that the vector quantization module of SLAM can significantly improve loop closure detection accuracy and recall. Time and space complexity, while using a deep convolutional autoencoder, are both linear and belong to $O(n)$. Moving forward, considering we have dataset of images over $1,000\ km$ path, to extend AE-FABMAP to operate in such trajectory deep autoencoders

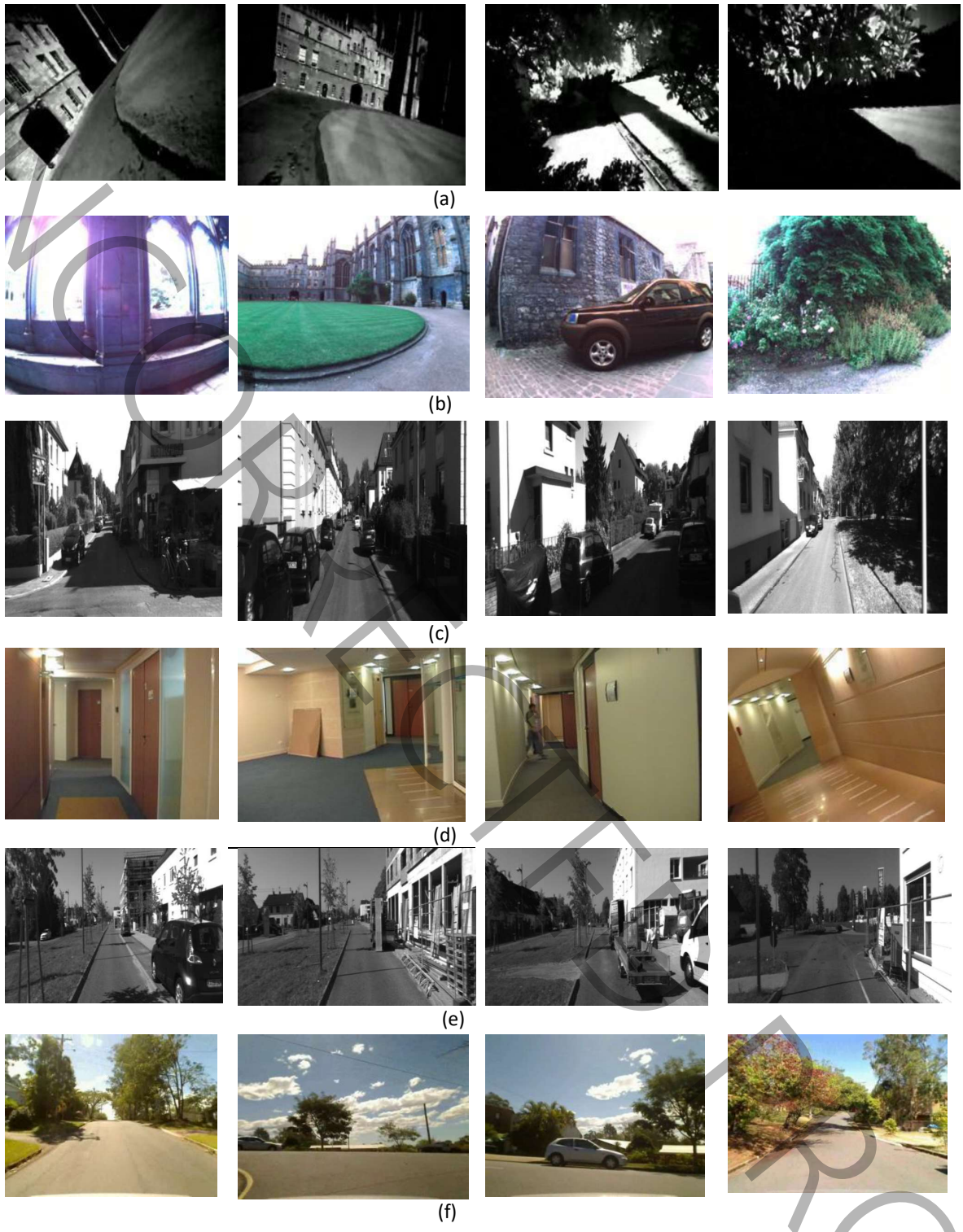

Figure 3: Datasets (a) Newer college, (b) New college, (c) Kitti sequence 7, (d) Lip6indoor, (e) KITTI sequence 6, (f) Stlucia (Tarin)

might give promising results. By using the architecture depicted in the figure 6 we aim to train an extensively large vocabulary (over $100,000$ centroids). By converting the data to several batches the local clustering and using the rough optimization function, (22) global clustering of image features will be achievable.

$$loss = \|\vec{x} - \vec{y}\| + \lambda_1 \times \|\vec{x_1} - \tilde{x_1}\| + \lambda_2 \times \|\vec{x_2} - \tilde{x_2}\| \tag{22}$$

Once we have the global clusters for such huge number of features we can perform a vector quantization over

UNCORRECTED PROOF

Table 3: Comparing AE-BoW, BoW, and HGCN-BoW methods for LCD.

Train Dataset	Test Dataset	AE-FABMAP		FABMAP2		HGCN-FABMAP	
		%Acc	%Rec	%Acc	%Rec	%Acc	%Rec
Stlucia 311417	Stlucia 111417	%83.38	%61.6	%83.6	%60	%94.1	%99.1
TUM seq-11 3924499	Lip6Indoor 27189	%82.4	%82.1	%46	%80	%62.8	%82.3
Kitti seq-6 227904	Kitti seq-7 235510	%60	%75.6	%71	%71.4	%80.2	
New College 160500	Newer College 194000	%71	%87	%67	%87	%51.1	%80.2

Table 4: Comparison between AE-ORB-SLAM and several state of the art trajectory generation methods

Method:	[37]	[13]	[44]	[53]	[15]	[33]	[26]
RMSE:	0.010	0.0473	0.054	0.1	0.3064	0.3853	0.5144

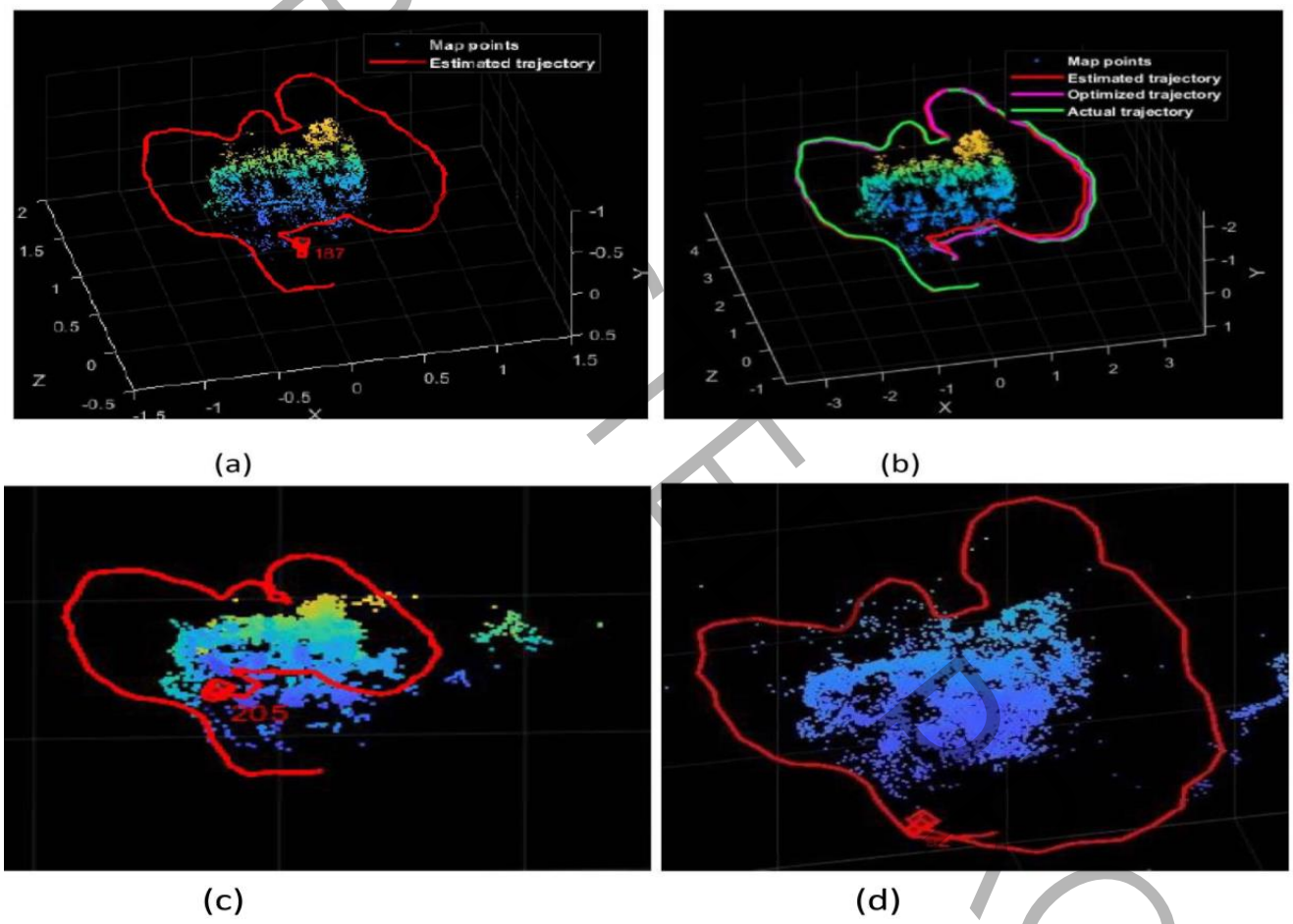

Figure 4: Comparison of the trajectory for Frieburg3 dataset using (a) ORB-SLAM, (b) Ground truth, (c) AE-ORB-SLAM and (d) HGCN-ORB-SLAM.

the extracted features. Next, we can feed the FABMAP and ORB-SLAM to obtain long range trajectory.

UNCORRECTED PROOF

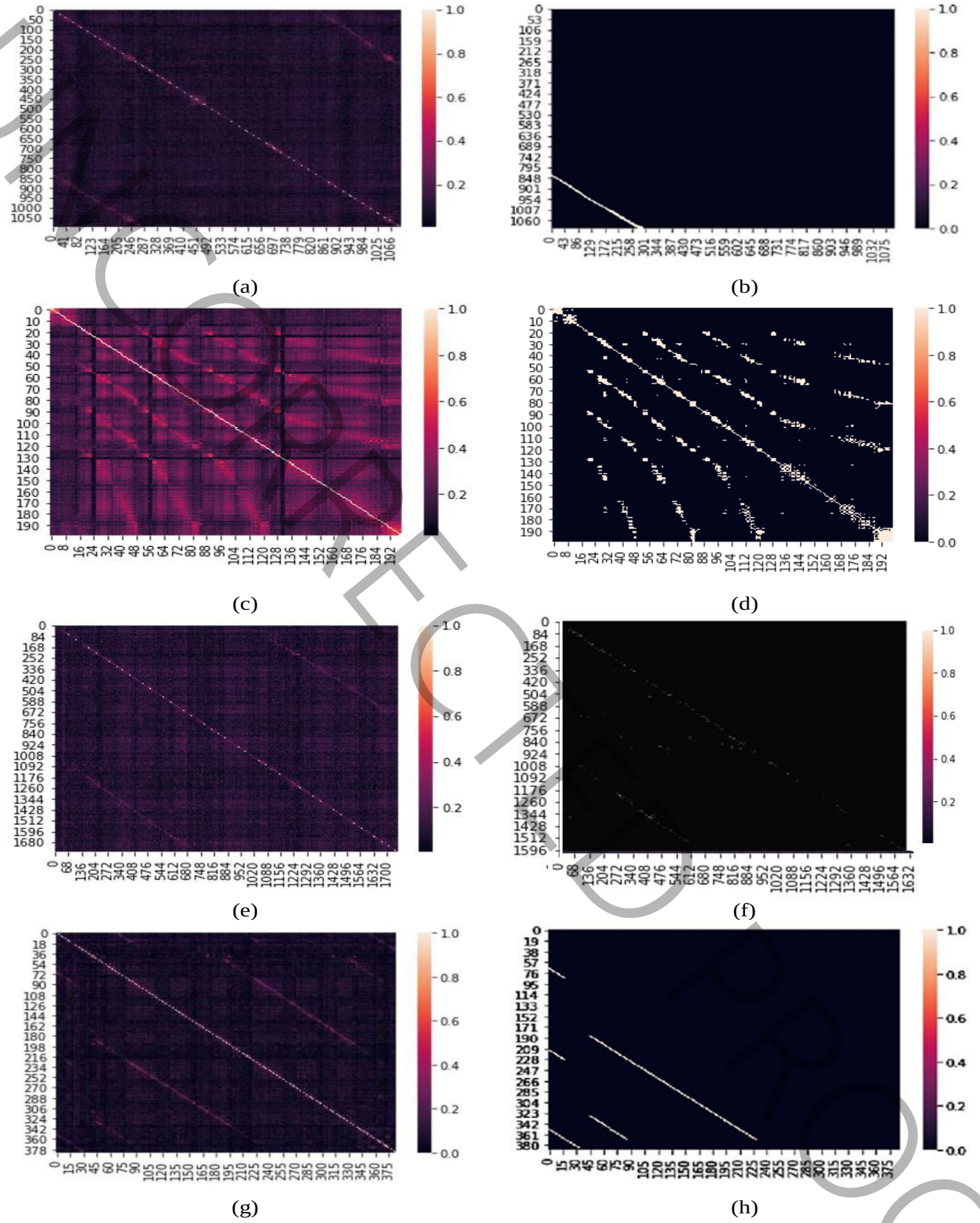

Figure 5: Confusion matrices for datasets (a) New-College, (b) Newer-College, (c) St Lucia, (d) Lip6indoor along with their corresponding ground truth matrices in (b), (d), (e), and (f).

6. Data Availability Statement

Publicly available image datasets were analyzed in this study such as dataset Lip6indoor which is publicly available in: `http://cogrob.ensta-paris.fr/assets/Lip6IndoorDataSet.tar.gz`. Other image datasets include Newer

UNCORRECTED PROOF

Table 5: Number of detected loop closures for datasets: Newer College, Saint-Lucia, Lip6-indoor, and Kitti 6th sequence

Dataset	Newer College	Saint Lucia	Lip6Indoor	Kitti 6^{th} seq
Number of total loops	3	1	5	1
Number of loops detected	3	1	5	1

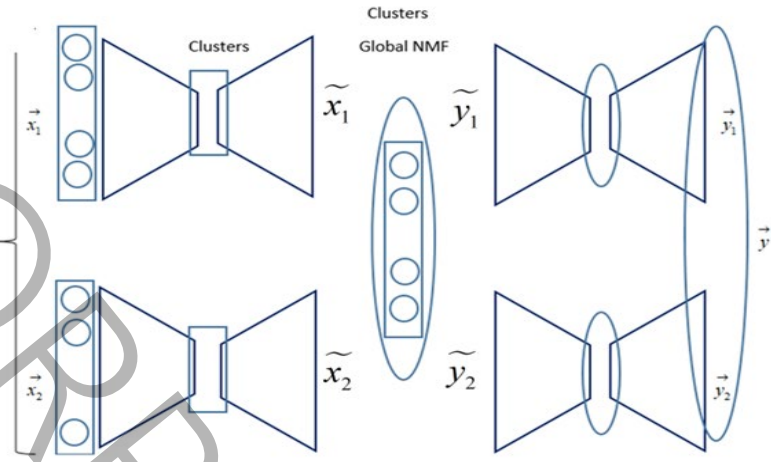

Figure 6: Auto-encoders chain to obtain global clustering.

College is a video (which we have converted into 200 image sequence), and it is publicly available here(Section quad with dynamics): `https://ori-drs.github.io/newer-college-dataset/stereo-cam/`, or it is directly available at `https://www.youtube.com/watch?v=sAmaJlql-vs&ab_channel=OxfordDynamicRobotSystemsGroup` Frieburg3 dataset used in AE-ORB SLAM and it is available on: `https://vision.in.tum.de/rgbd/dataset/freiburg3/rgbd_dataset_freiburg3_long_office_household.tgz`. The dataset TUM-sequence-11 is available online at `https://vision.in.tum.de/mono/dataset/sequence_11.zip` and datasets KITTI-6 and KITTI-7 are available online at Kitti Benchmark Suite: `https://www.cvlibs.net/datasets/kitti/user_login.php` (Email registration and conformation is required to get the datasets link).

Funding Statement

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1] M. Abouzahir, A. Elouardi, R. Latif, S. Bouaziz, and A. Tajer, *Embedding SLAM algorithms: Has it come of age?*, Robotics and Autonomous Systems, 100 (2018), pp. 14–26.

[2] S. Aldegheri, N. Bombieri, D. D. Bloisi, and A. Farinelli, *Data flow ORB-SLAM for real-time performance on embedded GPU boards*, in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 5370–5375.

[3] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software*, John Wiley & Sons, Inc., 2001.

[4] K. Boikos and C.-S. Bouganis, *A high-performance system-on-chip architecture for direct tracking for SLAM*, in 2017 27th International Conference on Field Programmable Logic and Applications (FPL), 2017, pp. 1–7.

[5] M. Bosse, P. Newman, J. Leonard, and S. Teller, *Slam in large-scale cyclic environments using the atlas framework*, International Journal of Robotic Research - IJRR, (2003).

[6] H. I. Christensen and O. Khatib, eds., *Robotics Research : The 15th International Symposium ISRR*, Springer International Publishing, 2017.

UNCORRECTED PROOF

[7] M. Cummins, *Probabilistic Localization and Mapping in Appearance Space*, PhD thesis, University of Oxford (United Kingdom), 2009.

[8] M. Cummins and P. Newman, *FAB-MAP: Probabilistic localization and mapping in the space of appearance*, The International Journal of Robotics Research, 27 (2008), pp. 647–665.

[9] A. Davison, *DSO: Direct sparse odometry*. Available online: `https://github.com/JakobEngel/dso`, accessed on 21 January 2022.

[10] ——, *Oxford-ptam*. Available online: `https://github.com/Oxford-PTAM/PTAM-GPL`, accessed on 21 January 2022.

[11] ——, *Scenelib 1.0. 2006*. Available online: `https://www.doc.ic.ac.uk/~ajd/Scene/index.html`, accessed on 21 January 2022.

[12] ——, *Svo*. Available online: `https://github.com/uzh-rpg/rpg_svo`, accessed on 21 January 2022.

[13] N. de Freitas and S. J. Godsill, *Rao-blackwellised particle filtering for dynamic bayesian networks*, in Advances in Neural Information Processing Systems, vol. 16, 2003, pp. 489–496.

[14] J. Engel, *LSD-SLAM: Large-scale direct monocular SLAM*. `https://github.com/tum-vision/lsd_slam`, accessed on 21 January 2022.

[15] J. Engel, T. Schöps, and D. Cremers, *Lsd-slam: Large-scale direct monocular SLAM*, in Computer Vision - ECCV 2014, Cham, 2014, Springer International Publishing, pp. 834–849.

[16] C. Estrada, J. Neira, and J. Tardos, *Hierarchical SLAM: real-time accurate mapping of large environments*, IEEE Transactions on Robotics, 21 (2005), pp. 588–596.

[17] C. Forster, M. Pizzoli, and D. Scaramuzza, *SVO: Fast semi-direct monocular visual odometry*, in 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 15–22.

[18] P. Foster, *OpenDTAM*. `https://github.com/anuranbaka/OpenDTAM`, accessed on 21 January 2022.

[19] D. Gálvez-López and J. D. Tardós, *Bags of binary words for fast place recognition in image sequences*, IEEE Transactions on Robotics, 28 (2012), pp. 1188–1197.

[20] E. Garcia-Fidalgo and A. Ortiz, *ibow-lcd: An appearance-based loop closure detection approach using incremental bags of binary words*, CoRR, abs/1802.05909 (2018).

[21] A. Geiger, P. Lenz, and R. Urtasun, *Are we ready for autonomous driving? the kitti vision benchmark suite*, in Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[22] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth, *OpenFABMAP: An open source toolbox for appearance-based loop closure detection*, in The International Conference on Robotics and Automation, Saint Paul, MN, USA, 2012, IEEE, pp. 4730–4735.

[23] K. Hajebi, *Efficient Visual Search in Appearance-based SLAM*, PhD thesis, University of Alberta, 2015.

[24] K. Hajebi and H. Zhang, *An efficient index for visual search in appearance-based SLAM*, in 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 353–358.

[25] A. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, *Visual odometry and mapping for autonomous flight using an RGB-D camera*, (2011).

[26] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, *Visual odometry and mapping for autonomous flight using an RGB-D camera*, in Christensen and Khatib [6].

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

[28] S. Khan and D. Wollherr, *Ibuild: Incremental bag of binary words for appearance based loop closure detection*, in 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 5441–5447.

[29] A. Kim and R. M. Eustice, *Combined visually and geometrically informative link hypothesis for pose-graph visual SLAM using bag-of-words*, in 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 1647–1654.

UNCORRECTED PROOF

[30] G. Klein and D. Murray, *Parallel tracking and mapping for small ar workspaces*, in 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007, pp. 225–234.

[31] ——, *Parallel tracking and mapping on a camera phone*, in 2009 8th IEEE International Symposium on Mixed and Augmented Reality, 2009, pp. 83–86.

[32] J. Leonard and P. Newman, *Consistent, convergent, and constant-time SLAM*, in IJCAI'03: Proceedings of the 18th international joint conference on Artificial intelligence, IJCAI'03, San Francisco, CA, USA, 2003, Morgan Kaufmann Publishers Inc., pp. 1143–1150.

[33] B. Liu, F. Tang, Y. Fu, Y. Yang, and Y. Wu, *A flexible and efficient loop closure detection based on motion knowledge*, in 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 11241–11247.

[34] A. Mostafanasab, M. B. Menhaj, M. Shamshirsaz, and R. Fesharakifard, *A novel mobile robot path planning method based on neuro-fuzzy controller*, AUT Journal of Mathematics and Computing, 6 (2025), pp. 41–53.

[35] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, *ORB-SLAM: A versatile and accurate monocular SLAM system*, IEEE Transactions on Robotics, 31 (2015), pp. 1147–1163.

[36] ——, *ORB-SLAM: a versatile and accurate monocular SLAM system*, (2015).

[37] R. Mur-Artal and J. D. Tardos, *ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras*, IEEE Transactions on Robotics, 33 (2017), pp. 1255–1262.

[38] P. Newman, M. Chandran-Ramesh, D. Cole, M. Cummins, A. Harrison, I. Posner, and D. Schroeter, *Describing, navigating and recognising urban spaces - building an end-to-end SLAM system*, in Proc. of the Int. Symposium of Robotics Research (ISRR), Hiroshima, Japan, 2007.

[39] P. Ondruska, P. Kohli, and S. Izadi, *Mobilefusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones*, IEEE Transactions on Visualization and Computer Graphics, 21 (2015), pp. 1251–1258.

[40] L. Paz, P. Jensfelt, J. Tardós, and J. Neira, *EKF SLAM updates in o(n) with divide and conquer SLAM*, in Proceedings 2007 IEEE International Conference on Robotics and Automation, IEEE International Conference on Robotics and Automation ICRA, 2007, pp. 1657–1663.

[41] T. Pire, T. Fischer, G. Castro, P. D. Cristóforis, J. Civera, and J. J. Berlles, *S-ptam: Stereo parallel tracking and mapping*, Robotics and Autonomous Systems, 93 (2017), pp. 27–42.

[42] S. Se, D. G. Lowe, and J. Little, *Vision-based global localization and mapping for mobile robots*, IEEE Transactions on Robotics, 21 (2005), pp. 364–375.

[43] J. Soares and M. Meggiolaro, *Keyframe-based RGB-D SLAM for mobile robots with visual odometry in indoor environments using graph optimization*, in 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE), 2018, pp. 94–99.

[44] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, *A benchmark for the evaluation of rgb-d SLAM systems*, in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 573–580.

[45] V. Sundar, *CNN SLAM*. Available online: `https://github.com/iitmcvg/CNN_SLAM`, accessed on 21 January 2022.

[46] V. Turchenko, E. Chalmers, and A. Luczak, *A deep convolutional auto-encoder with pooling - unpooling layers in caffe*, CoRR, abs/1701.04949 (2017).

[47] V. Turchenko, E. Chalmers, and A. Luczak, *A deep convolutional auto-encoder with pooling - unpooling layers in caffe*, International Journal of Computing, 18 (2019), pp. 8–31.

[48] B. Vincke, A. Elouardi, and A. Lambert, *Design and evaluation of an embedded system based SLAM applications*, in 2010 IEEE/SICE International Symposium on System Integration, 2010, pp. 224–229.

UNCORRECTED PROOF

[49] B. Vincke, A. Elouardi, A. Lambert, and A. Merigot, *Efficient implementation of EKF-SLAM on a multi-core embedded system*, in IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, 2012, pp. 3049–3054.

[50] T. Ying, H. Yan, Z. Li, K. Shi, and X. Feng, *Loop closure detection based on image covariance matrix matching for visual SLAM*, International Journal of Control, Automation and Systems, 19 (2021).

[51] J. Yu, F. Gao, J. Cao, C. Yu, Z. Zhang, Z. Huang, Y. Wang, and H. Yang, *CNN-based monocular decentralized SLAM on embedded fpga*, in 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2020, pp. 66–73.

[52] A. Zarringhalam, *AE-FABMAP*. `https://github.com/amir-1992/AE-FABMAP-SLAM`, 2024.

[53] ——, *AE-ORB*. `https://github.com/amir-1992/AE-ORB-SLAM`, 2024.

[54] A. Zarringhalam, S. S. Ghidary, and A. M. Khorasani, *Self-supervised vector-quantization in visual SLAM using deep convolutional autoencoders*, 2022.

[55] A. Zarringhalam, S. Shiry Ghidary, A. Mohades, and S.-A. Sadegh-Zadeh, *CUDA and OpenMp implementation of boolean matrix product with applications in visual SLAM*, Algorithms, 16 (2023).

[56] A. Zarringhalam, S. Shiry Ghidary, A. Mohades, and S.-A. Sadegh-Zadeh, *Semisupervised vector quantization in visual SLAM using hgcn*, International Journal of Intelligent Systems, 2024 (2024), p. 9992159.

[57] Q. Zhong and X. Fang, *A BigBiGAN-based loop closure detection algorithm for indoor visual SLAM*, Journal of Electrical and Computer Engineering, 2021 (2021).

Appendix

```
#include "inference.hpp"

namespace of2 {

    int InferBase::pq(int q) {
        return (int)\text{CLtree}.at<double>(0,q);
    }

    double InferBase::Pzq(int q, bool zq) {
        return (zq) ? \text{CLtree}.at<double>(1,q) : 1 - \text{CLtree}.at<double>(1,q);
    }

    double InferBase::PzqGzpq(int q, bool zq, bool zpq) {
        if (zpq) {
            return (zq) ? \text{CLtree}.at<double>(2,q) : 1 - \text{CLtree}.at<double>(2,q);
        } else {
            return (zq) ? \text{CLtree}.at<double>(3,q) : 1 - \text{CLtree}.at<double>(3,q);
        }
    }

    double InferBinary::PzqGeq(bool zq, bool eq) {
        if (eq) {
            return (zq) ? PzGe : 1 - PzGe;
        } else {
            return (zq) ? PzGNe : 1 - PzGNe;
        }
    }

    double InferBinary::PeqGLzq(int q, bool Lzq, bool eq) {
        double alpha, beta;
        alpha = PzqGeq(Lzq, true) * Pzq(q, true);
        beta = PzqGeq(Lzq, false) * Pzq(q, false);

        if (eq) {
            return alpha / (alpha + beta);
        } else {
            return 1 - (alpha / (alpha + beta));
        }
    }

```

UNCORRECTED PROOF

```
        double InferBinary::PzqGL(int q, bool zq, bool /*zpq*/, bool Lzq,
        const bool & newPlace /*= false*/)
        {
            double p = (newPlace ? Pzq(q, false) : PeqGLzq(q, Lzq, false)) * PzqGeq(zq, false) +
            (newPlace ? Pzq(q, true) : PeqGLzq(q, Lzq, true)) * PzqGeq(zq, true);

            return p;
        }

        double InferBinary::PzqGzpqL(int q, bool zq, bool zpq, bool Lzq,
        const bool & newPlace /*= false*/) {
            double p;
            double alpha, beta;

            alpha = Pzq(q,  zq) * PzqGeq(!zq, false) * PzqGzpq(q, !zq, zpq);
            beta  = Pzq(q, !zq) * PzqGeq( zq, false) * PzqGzpq(q,  zq, zpq);
            p = (newPlace ? Pzq(q, false) : PeqGLzq(q, Lzq, false))
            * beta / (alpha + beta);

            alpha = Pzq(q,  zq) * PzqGeq(!zq, true) * PzqGzpq(q, !zq, zpq);
            beta  = Pzq(q, !zq) * PzqGeq( zq, true) * PzqGzpq(q,  zq, zpq);
            p += (newPlace ? Pzq(q, true) : PeqGLzq(q, Lzq, true))
            * beta / (alpha + beta);

            return p;
        }

    } // namespace of2
```

Please cite this article using:

Amir Zarringhalam, Ali Mohades, Saeed Shiry Ghidary, Loop closure detection in visual appearance-based SLAM using deep autoencoders, AUT J. Math. Comput., 7(2) (2026) 117-135
https://doi.org/10.22060/AJMC.2024.23054.1224

UNCORRECTED PROOF