Adaptive Soft Encoding: A General Unsupervised Feature Aggregation Method for Place Recognition

Yansong Gong, Jing Yuan, Fengchi Sun, Qinxuan Sun, Wenbin Zhu, and Xuebo Zhang, *Senior Member, IEEE*

***Abstract*— Place recognition (PR) is an important problem in environment perception, which can help simultaneous localization and mapping (SLAM), as well as scene understanding of mobile robots. For PR, feature aggregation plays an important role in the representation of environments. In this article, an adaptive soft encoding (ASE) method is proposed for aggregating numerous features into low-dimensional feature vectors, which includes a training phase and an encoding phase. For the training phase, the information along each dimension of the features is evaluated, which is further employed to assign all dimensions into different subdimensional intervals. After that, subdimensional Gaussian mixture models (SD-GMMs) are fit by features in the subdimensional intervals and organized into an ASE tree, of which the tree nodes hierarchically store the parameters of the SD-GMMs to reflect the distribution of the features. For the encoding phase, features are fed into the root node of the ASE tree. The weight of each node is calculated from top to bottom to generate multiple aggregated feature vectors with different description capabilities. In addition, an ASE-based framework for PR is proposed, in which local descriptors are extracted from sensor data (e.g., images or LiDAR data), and then aggregated into global descriptors by ASE. Finally, owing to the hierarchical structure of the ASE tree, a hierarchical matching strategy of the global descriptors is designed to recognize places efficiently. Experimental results demonstrate that feature vectors aggregated by ASE have strong distinguishability, and the ASE-based PR method achieves good accuracy and efficiency. The code of ASE can be accessed at https://github.com/wdyiwdwd/ASE-Encoder.**

***Index Terms*— Adaptive soft encoding (ASE), feature aggregation, mobile robots, place recognition (PR).**

NOMENCLATURE

D	Set of features, $D = \{\boldsymbol{d_i}, i = 1, 2, \ldots, n\}$.
$\boldsymbol{d_i}$	ith feature in D, $\boldsymbol{d_i} = [d_{i,1}, d_{i,2}, \ldots, d_{i,m}]^T \in \mathbb{R}^m$.
n_k	Number of bins divided along the kth dimension in the soft encoding (SE).
δ_k	Resolution of bins along the kth dimension in SE.
$\omega_{i,k,p}$	Weights measuring how close $d_{i,k}$ is to its bin boundaries in SE.
$\boldsymbol{f}$	Feature vector aggregated by SE, $\boldsymbol{f} = [f_1, f_2, \ldots, f_{n_f}]^T \in \mathbb{R}^{n_f}$.
$\mathcal{X}$	Information amounts along each dimension in adaptive SE (ASE), $\mathcal{X} = \{\chi_k, k = 1, 2, \ldots, m'\}$.
N	Number of subdimensional intervals in ASE.
$\mathcal{Z}$	Subdimensional interval boundaries in ASE $\mathcal{Z} = \{\zeta_i, i = 0, 1, \ldots, N\}$.
$\mathcal{G}_i$	SD-GMM corresponding to the ith subdimensional intervals in ASE.
$\mathcal{T}_{i,j}$	jth tree node in the ith level of the ASE tree.
$\kappa_{i,j}$, $\iota_{i,j}$	Intermediate results to calculate aggregated feature vectors in $\mathcal{T}_{i,j}$.
$\mathcal{F}$	Set of aggregated feature vectors in the ASE tree, $\mathcal{F} = \{\boldsymbol{f}_j, j = 1, 2, \ldots, N\}$.
$\mathcal{P}$	Place.
$\mathcal{L}_\mathcal{P}$	Set of local descriptors extracted from the sensor data collected in $\mathcal{P}$.
$\eth_\mathcal{P}$	Set of global descriptors aggregated by ASE to represent $\mathcal{P}$, $\eth_\mathcal{P} = \{\boldsymbol{g}^i_\mathcal{P}, i = 1, 2, \ldots, N\}$.
$\boldsymbol{g}^i_\mathcal{P}$	ith global descriptor aggregated by ASE, corresponding to the feature vector in the ith level of the ASE tree.
γ_i	Information amounts contained in $g^i_\mathcal{P}$.

Manuscript received 8 May 2023; revised 27 July 2023; accepted 5 August 2023. Date of publication 23 August 2023; date of current version 7 September 2023. This work was supported in part by the Natural Science Foundation of China under Grant U21A20486, Grant 62073178, and Grant 61873327; in part by the Tianjin Science Fund for Distinguished Young Scholars under Grant 20JCJQJC00140; and in part by the Tianjin Natural Science Foundation under Grant 20JCYBJC01470. The Associate Editor coordinating the review process was Dr. Qing Wang. *(Corresponding authors: Jing Yuan; Fengchi Sun.)*

Yansong Gong and Fengchi Sun are with the College of Software, Nankai University, Tianjin 300350, China (e-mail: fengchisun@nankai.edu.cn).

Jing Yuan, Wenbin Zhu, and Xuebo Zhang are with the College of Artificial Intelligence, Nankai University, Tianjin 300350, China (e-mail: nkyuanjing@gmail.com).

Qinxuan Sun is with the Beijing Research Institute, Zhejiang Lab, Beijing 100086, China.

Digital Object Identifier 10.1109/TIM.2023.3307752

I. INTRODUCTION

PLACE recognition [1], [2], [3], [4] is an important task in the field of environment perception. A good method of place recognition (PR) can correctly and efficiently recognize a place by matching current sensor data with historical data, which is helpful in terrain assessment [5], scene understanding [6], and localization of mobile robots [7]. In addition, PR can be used in simultaneous localization and mapping (SLAM) [8], [9], [10] to detect a loop closure and further to eliminate accumulated errors [11].

Feature aggregation refers to the process of combining high-dimensional features into a feature vector. Popular methods of feature aggregation are based on by clustering techniques, including bag of words (BoW) [12] and vector of locally aggregated descriptors (VLAD) [13]. These methods allow for efficient modeling of complex data by reducing the

dimensionality of the feature space while preserving important information. Feature aggregation plays an important role in PR. It relies on aggregating high-dimensional features of sensor data into feature vectors to describe places, which are matched to recognize places by similarities between the feature vectors [14].

Feature aggregation methods are usually utilized in local descriptor-based PR, in which local descriptors are extracted from sensor data to describe a place, e.g., the speeded-up robust features (SURFs) [15], scale-invariant feature transforms (SIFTs) [16], and oriented fast and rotated briefs (ORBs) [17] from images, signatures of histogram of orientation (SHOTs) [18], and point feature histograms (PFHs) [19] from LiDAR data. Compared with global descriptor-based methods [20], [21], [22], [23], [24], [25], [26], local descriptors are more robust to the angle of view of the sensors. However, establishing correspondences between numerous local descriptors in different places is inefficient. Therefore, feature aggregation is utilized to aggregate local descriptors into global descriptors for representing environments, which takes advantages of both methods.

For instance, the fast appearance-based mapping (FAB-MAP) [14] extracted SURFs from images, and a feature aggregation method, probability-based BoW [12], was applied to encode the obtained SURFs into a binary global descriptor. In [27], the echo state network (ESN) was applied to model image sequences. Then, the position-invariant robust features (PIRFs) were extracted from the image sequences and aggregated into a global descriptor by a BoW [12] approach. Likewise, in [28], normal aligned radial features (NARFs) were extracted from LiDAR data and aggregated into a global descriptor by BoW. The PointNetVLAD [29] utilized a PointNet to extract features from LiDAR data collected in a local area, and the obtained features were aggregated by a supervised feature aggregation method, i.e., NetVLAD. In [30], the spatial relation descriptors (SRDs) were integrated into a graph-based representation called spatial relation graph (SRG), and an increment BoW model [31] was utilized to retrieve similar SRGs from historical data. In our previous studies [11], [32], structural units were extracted from 2-D and 3-D LiDAR data (2-D SUs and 3-D SUs) and aggregated into global descriptors by SE.

Feature aggregation methods in the field of PR can be categorized into two types: supervised methods [29] and unsupervised methods [11], [12], [13]. Supervised methods require labeled data for pretraining, which takes a long period to be deployed and usually requires GPU acceleration, which limits their applications.

Unsupervised methods perform feature aggregation adaptively according to the distribution of input features, which are divided into clustering-based methods and statistic-based methods. Clustering-based methods [12], [13] cluster features and generate a feature vector based on the relationship between the clusters of features. For instance, BoW clusters the features using the K-means algorithm and aggregates the features by counting the number of features in each cluster. However, clustering-based methods cannot distinguish data accurately at cluster boundaries, as they lack a probability-based data model, and thus, are not robust to the noise. In addition, due to clustering, these methods incur high-computational complexity, resulting in lower training efficiency.

Different from clustering-based methods, statistic-based methods [11], [32], [33] do not require the pretraining process. The histogram [33] was a representative statistic-based method. It was generated by counting the number of feature element in each bin, which was manually set. In order to improve the performance of the histogram, the SE [32] not only considered to which bin the feature element belonged, but also calculated weights to reflect to what extent the element belonged to the corresponding bin. Statistic-based methods are fast and easy to implement. For low-dimensional features, results of statistic-based methods usually have high distinguishability [32]. However, since statistic-based methods consider the distribution on features along each dimension, the length of the aggregated vector increases greatly, as the dimension of features increases. Therefore, statistic-based methods are not well suited for high-dimensional features.

In this article, a novel feature aggregation method called ASE is proposed, which provides SE with the adaptability through an unsupervised training phase and takes advantages from both clustering-based and statistic-based methods. In ASE, features are first normalized and orthogonalized. Then, the amount of information along each dimension of features is analyzed, and subdimensional intervals are assigned to make the information amount of each subdimensional interval similar. Next, subdimensional Gaussian mixture models (SD-GMMs) are fit according to the distribution of features in each subdimensional interval. After that, an ASE tree is established to organize SD-GMMs and utilized for aggregating features into a set of feature vectors with low dimensions. Compared with clustering-based methods, ASE is more efficient in the training phase. Compared with statistic-based methods, ASE is adaptive to high-dimensional features.

In addition, an ASE-based PR framework is proposed in this article. First, local descriptors, such as ORBs and SIFTs from images or SRDs and 3-D SUs from LiDAR data, are extracted. Then, a small amount of local descriptors are used for fast and online training, which generates an ASE tree. Then, the local descriptors are aggregated into multiple global descriptors by the ASE tree. These global descriptors of a place contain different information amounts and have different description capabilities. Finally, PR is performed by hierarchical matching of these global descriptors. The contributions of this article are summarized as follows.

1) A novel unsupervised feature aggregation method, ASE, is proposed, which improves SE with adaptability to features with any types and dimensions. Moreover, the ASE method only needs a small amount of training data and can quickly generate an effective data representation model without the need of GPU acceleration.
2) An ASE-based PR framework is proposed, in which different types of local descriptors from different types of sensors can be aggregated into distinguishable global descriptors. Then, a hierarchical matching strategy is designed, which utilizes global descriptors with different information amounts and lengths to accelerate the PR.

3) Extensive experiments are carried out on public and self-built data sequences collected by 3-D LiDARs and cameras, demonstrating that ASE and the ASE-based PR framework achieve good accuracy and efficiency.

The rest of this article is organized as follows. The system overview is presented in Section II. In Section III, we give a detailed description for ASE. The ASE-based PR framework is introduced in Section IV. Comprehensive experimental evaluations are shown in Section V. The conclusions are drawn in Section VI.

II. System Overview

In this article, the ASE-based framework is designed to describe a place by aggregating local descriptors extracted from sensor data into global descriptors and match global descriptors to recognize a place. The overview of the ASE-based framework for PR is shown in Fig. 1, which is composed of two phases, i.e., description and search.

In the description phase, local descriptors are first extracted from sensor data (e.g., images and LiDAR data), yielding local descriptor set $\mathcal{L}$. Then, local descriptors in the first K_d frames are utilized to perform ASE training, including four steps, i.e., preprocessing model training, subdimensional interval dividing, SD-GMM fitting, and ASE tree construction. For ASE training, the information amount along each dimension of local descriptors is evaluated, and then, dimensions of local descriptors are assigned into different subdimensional intervals according to information amounts. For each subdimensional interval, a GMM is utilized to fit the distribution of local descriptors in this subdimensional interval, called SD-GMM. After that, SD-GMMs in all subdimensional intervals are organized into a tree structure called ASE tree, which is utilized for calculating global descriptors from top to bottom.

For ASE encoding, local descriptors obtained in the description phase are first preprocessed and then fed into the root node of the ASE tree to aggregate them into a set of global descriptors $\eth = \{\boldsymbol{g_1}, \boldsymbol{g_2}, \ldots, \boldsymbol{g_N}\}$. It is worth pointing out that, for a set of local descriptors, multiple global descriptors can be aggregated by ASE to characterize a place. They contain information from less to more, indicating that they describe a place from coarse to fine. Specifically, $\boldsymbol{g_i}$ is generated from $\boldsymbol{g_{i-1}}$, and as i gets larger, $\boldsymbol{g_i}$ becomes longer. The longer the $\boldsymbol{g_i}$ is, the more the information $\boldsymbol{g_i}$ contains.

In the search phase, a hierarchical matching strategy is proposed for matching global descriptors of places from coarse to fine. In order to improve the matching efficiency, the historical places, which have significant appearance differences with the current place, are filtered out in stages by matching global descriptors with different lengths and information amounts. The notation used throughout this article is listed in Nomenclature.

III. Adaptive Soft Encoding

A. Soft Encoding

SE proposed in our previous work [32] is a feature aggregation scheme for aggregating SUs extracted from 2-D LiDAR data. However, in [32], the input features are limited

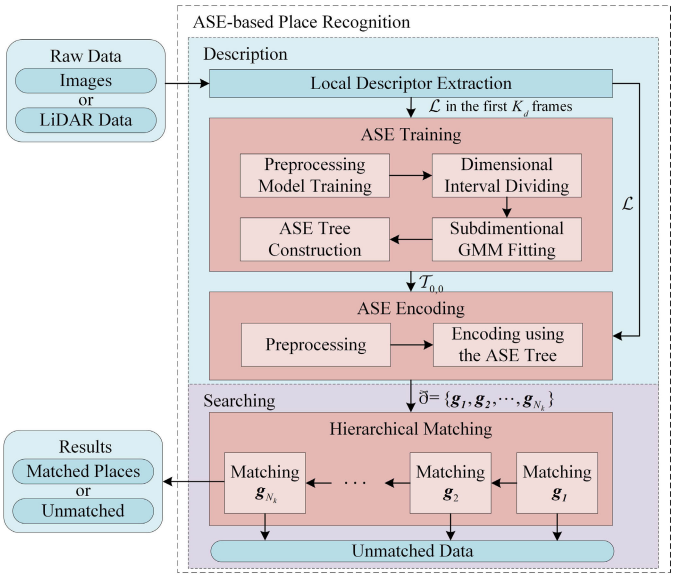


Fig. 1. Architecture overview of the ASE-based framework for PR.

to a specific form of three dimensions, which indicates that no general expression of SE is given for features with any dimension. In this section, the original SE is extended to any m-dimensional ($m \in \mathbb{N}$) features. The input feature set is denoted by $D = \{\boldsymbol{d_i}, i = 1, 2, \ldots, n\}$, where $\boldsymbol{d_i} = [d_{i,1}, d_{i,2}, \ldots, d_{i,m}]^T \in \mathbb{R}^m$. The aggregated feature vector of D is denoted by $\boldsymbol{f} = [f_1, f_2, \ldots, f_{n_f}]^T \in \mathbb{R}^{n_f}$. The procedure of SE includes two steps, i.e., weight calculating and weight combining, which are illustrated in Fig. 2, and the overall procedure is summarized in Algorithm 1.

For weight calculating, along the kth dimension of features in D, two preset thresholds $d_k^{\min}$ and $d_k^{\max}$ are given according to the prior information, which are the lower and upper bounds, respectively. Then, the kth dimension of features is divided into n_k bins with the resolution calculated by $\delta_k = (d_k^{\max} - d_k^{\min})/n_k$ and the dividing boundaries by $l_{k,j} = d_k^{\min} + j \cdot \delta_k$, $j = 0, 1, \ldots, n_k$. Next, the index of the bin along the kth dimension to which $\boldsymbol{d_i} \in D$ belongs is calculated by

$$\eta_{i,k} = \left\lfloor \frac{d_{i,k} - d_k^{\min}}{\delta_k} \right\rfloor. \tag{1}$$

The weights $\omega_{i,k,1}$ and $\omega_{i,k,2}$ measuring how close $d_{i,k}$ is to the dividing boundaries $l_{k,\eta_{i,k}}$ and $l_{k,\eta_{i,k}+1}$ are calculated by

$$\begin{cases} \omega_{i,k,2} = \left(d_{i,k} - l_{k,\eta_{i,k}}\right)/\delta_k \\ \omega_{i,k,1} = \left(l_{k,\eta_{i,k}+1} - d_{i,k}\right)/\delta_k. \end{cases} \tag{2}$$

From (2), two weights satisfy $\omega_{i,k,1} + \omega_{i,k,2} = 1$, $k = 1, 2, \ldots, m$.

For weight combining, for $\boldsymbol{d_i} \in \mathbb{R}^m$, m pairs of weights used to calculate elements in $\boldsymbol{f}$ are calculated and combined. Above all, an index–value pair is defined by $\psi = (\tau, \rho)$, in which τ represents the index of ρ and ρ represents an element of the resultant feature vector generated from a single feature in D. Then, the weights calculated by $d_{i,k}$ ($k = 1, 2, \ldots, m$) are multiplied continuously, denoted by $\rho = \prod_{k=1}^{m} \omega_{i,k,p_k}$ ($p_k = 1$ or 2). The corresponding element index τ of the aggregated

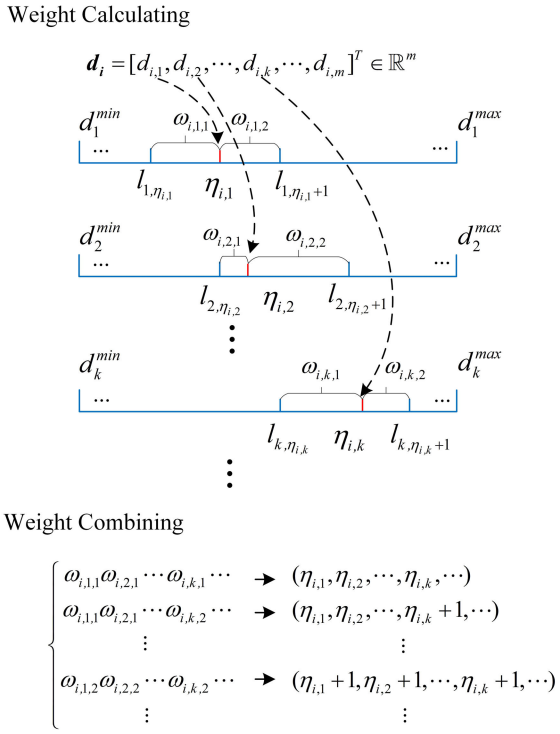


Fig. 2. Procedure of SE, including weight calculating and weight combining. For weight calculating, for each element $d_{i,k}$ in $\boldsymbol{d}_i \in D$, the index $\eta_{i,k}$ of the bin along the kth dimension and the weights $\omega_{i,k,1}, \omega_{i,k,2}$ that measure proximity to its bin boundaries are calculated. For weight combining, the weights from different dimensions are multiplied sequentially, and the obtained value represents an element of the aggregated feature vector. The index of the element is calculated by the bin index $\eta_{i,k}$, as given in (3).

feature vector is calculated by the bin index $\eta_{i,k}$. Specifically, τ is calculated by a linear combination of $\eta_{i,k}$

$$\tau = 1 + \sum_{k=1}^{m} \left(\left(\prod_{q=k+1}^{m} (n_q + 1) \right) \cdot \eta_k \right) \tag{3}$$

where

$$\eta_k = \begin{cases} \eta_{i,k}, & \text{if } p_k = 1 \\ \eta_{i,k} + 1, & \text{if } p_k = 2. \end{cases} \tag{4}$$

The length n_f of $\boldsymbol{f}$ is calculated by $\prod_{k=1}^{m} (n_k + 1)$. Meanwhile, since the two weights are calculated by a 1-D element, there are 2^m combinations of weights in total (e.g., [32, Table I] shows all the combinations and the corresponding indices when aggregating 3-D features). Then, the τth element in $\boldsymbol{f}$ is added by ρ/n for each combination. Next, the aggregated feature vector $\boldsymbol{f}$ of D is calculated by accumulating results of all features in D. Finally, $\boldsymbol{f}$ is normalized by $L2$ normalization.

Compared with the traditional statistic-based method, SE makes a better fit to the feature distribution in each dimension and thus, has better distinguishability. However, SE still has three main limitations, which restrict its application in PR with different types of input features.

1) In SE, two preset thresholds $d_k^{\min}$ and $d_k^{\max}$ are required for dividing bins. However, for most local descriptors extracted from images or LiDAR data, determining these thresholds a priori is difficult, because the elements in these descriptors might not have physical meanings, such as ORBs or SIFTs. In addition, considerable parameters to be set manually make SE difficult to tune in complex situations.
2) SE is not adaptive to high-dimensional features. For features with m dimensions, if k bins are counted for each dimension, the aggregated vector is in $\mathbb{R}^{(k+1)^m}$. As the dimension increases, the length of the vector increases exponentially, which suggests that SE is unsuitable for high-dimensional features.
3) SE encodes the element in each dimension equally, neglecting that elements in different dimensions may be strongly relevant or contain different amounts of information. Therefore, feature vectors aggregated by SE may contain much redundant information.

Algorithm 1 Soft Encoding

Input: a set of features $D = \{\boldsymbol{d}_i, i = 1, 2, \cdots, n\}$, where $\boldsymbol{d}_i = [d_{i,1}, d_{i,2}, \cdots, d_{i,m}]^T \in \mathbb{R}^m$. The range of features in each dimension and the number of dividing intervals are given manually, denoted by $R = \{\{d_k^{max}, d_k^{min}\}, k = 1, 2, \cdots, m\}$ and $N = \{n_k, k = 1, 2, \cdots, m\}$, respectively.
Output: an aggregated feature vector $\boldsymbol{f} = \{f_p, p = 1, 2, \cdots, \prod_{i=1}^{m} (n_i + 1)\}$ to describe D.

for $q = 1 : \prod_{i=1}^{m} (n_k + 1)$ **do**
  $f_q = 0$
end for
for $k = 1 : m$ **do**
  $\delta_k = (d_k^{max} - d_k^{min})/n_k$
  $L_k = \{l_{k,j} = d_k^{min} + j \cdot \delta_k, j = 0, 1, \cdots, n_k\}$
end for
for $\boldsymbol{d}_i$ in D **do**
  for $d_{i,k}$ in $\boldsymbol{d}_i$ **do**
    $\eta_{i,k} = \lfloor (d_{i,k} - d_k^{min})/\delta_k \rfloor$
    $\omega_{i,k,2} = \left(d_{i,k} - l_{k,\eta_{i,k}}\right)/\delta_k$
    $\omega_{i,k,1} = \left(l_{k,\eta_{i,k}+1} - d_{i,k}\right)/\delta_k$
  end for
  $\Gamma = \{\psi_0 = (\tau_0, \rho_0)\}$, where $\tau_0 = 1$ and $\rho_0 = 1.0$
  for $k = 1 : m$ **do**
    $\Gamma' = \emptyset$
    for $\psi = (\tau, \rho)$ in Γ **do**
      $\tau_1 = \tau + (\prod_{q=k+1}^{m} (n_q + 1)) \cdot \eta_{i,k}$, $\rho_1 = \rho \cdot \omega_{i,k,1}$
      $\tau_2 = \tau + (\prod_{q=k+1}^{m} (n_q + 1)) \cdot (\eta_{i,k} + 1)$, $\rho_2 = \rho \cdot \omega_{i,k,2}$
      $\psi_1 = (\tau_1, \rho_1)$, $\psi_2 = (\tau_2, \rho_2)$
      $\Gamma' = \Gamma' \cup \{\psi_1, \psi_2\}$
    end for
    $\Gamma = \Gamma'$
  end for
  for $\psi = (\tau, \rho)$ in Γ **do**
    $f_\tau = f_\tau + \rho/n$
  end for
end for
Compute L-2 normalization for $\boldsymbol{f}$

B. Adaptive Soft Encoding

ASE is an improvement of SE to provide it with adaptability. ASE can aggregate any type of features with any dimension and improve the distinguishability of SE. Specifically, such as SE, ASE still utilizes combinations of weights to aggregate features. However, ASE and SE are quite different in dimension selection, weight calculation, and result organization. In general, ASE includes two phases, i.e., the training phase and the encoding phase. The training phase include four steps, i.e., *preprocessing model training*, *subdimensional interval dividing*, *SD-GMM fitting*, and *ASE tree construction*. The encoding phase contains two steps, including *preprocessing* and *encoding using the ASE tree*. In comparison, the two steps of SE, i.e., weight calculating and weight combining, are both included in the last step *encoding using the ASE tree*, in which the ASE tree calculates and combines weights adaptively, without the need to set the boundaries manually.

1) Training:

a) Preprocessing model training: In ASE, input features are normalized and orthogonalized using z-score and the principal component analysis (PCA), respectively. In the training phase, the parameters of z-score and PCA need to be learned according to the distribution of features. Given a set of features for training, denoted by $D^{\text{tr}} = \{\boldsymbol{d}_i^{\text{tr}}, i = 1, 2, \ldots, n^{\text{tr}}\}$, where $\boldsymbol{d}_i^{\text{tr}} = [d_{i,1}^{\text{tr}}, d_{i,2}^{\text{tr}}, \ldots, d_{i,m}^{\text{tr}}]^T \in \mathbb{R}^m$, μ_j and σ_j are calculated to represent the average and standard deviation of n^{tr} number of features along the jth dimension. Then, the jth element in $\boldsymbol{d}_i^{\text{tr}}$ is normalized by

$$d_{i,j}^{\text{tr}'} = \frac{d_{i,j}^{\text{tr}} - \mu_j}{\sigma_j}. \tag{5}$$

The normalized set of D^{tr} is denoted by $D^{\text{tr}'}$. For PCA in the training phase, the covariance matrix $\boldsymbol{C}^{\text{tr}}$ of $D^{\text{tr}'}$ is first calculated. Then, the eigendecomposition of $\boldsymbol{C}^{\text{tr}}$ is calculated by $\boldsymbol{C}^{\text{tr}} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}$, where $\boldsymbol{\Lambda} = \text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ and $\boldsymbol{Q} = [\boldsymbol{q_1}, \boldsymbol{q_2}, \ldots, \boldsymbol{q_m}]^T$, representing eigenvalues and eigenvectors of $\boldsymbol{C}^{\text{tr}}$, respectively. Finally, the first m' rows of $\boldsymbol{Q}^T$ are selected and denoted by $\boldsymbol{Q}^{T'}$, where the first m' eigenvalues account for more than 95% of the total eigenvalue. Then, $\boldsymbol{d}_i^{\text{tr}'} \in D^{\text{tr}'}$ is orthogonalized by $\boldsymbol{d}_i^{\boldsymbol{t}} = \boldsymbol{Q}^{T'} \cdot \boldsymbol{d}_i^{\text{tr}'}$. In this way, dimension reduction and decorrelation of training features can be achieved, and the preprocessed feature set is obtained, denoted by $D^t = \{\boldsymbol{d}_i^{\boldsymbol{t}}, i = 1, 2, \ldots, n^{\text{tr}}\}$.

b) Subdimensional interval dividing: In SE, the elements of features in every dimension are utilized to calculate weights, making aggregated vectors redundant. In ASE, in order to reduce the redundancy, weights are calculated according to the distribution of features in a subdimensional interval instead of every dimension. First, the amount of information along each dimension is analyzed in ASE. Then, N subdimensional intervals are divided, making the information amount in each subdimensional interval similar, as shown in Fig. 3. The process of subdimensional interval dividing is summarized in Algorithm 2.

First, two functions in Algorithm 2 are introduced. The function $\text{BLOCK}(\boldsymbol{d}, \zeta_i, \zeta_j)$ in lines 1–4 of Algorithm 2 is to block elements in $\boldsymbol{d}$ with the indices from ζ_i to ζ_j to generate a new subfeature $\boldsymbol{d}^{\boldsymbol{s}}$. The function $\text{DIVIDE}(D, \zeta_i, \zeta_j)$ in lines 5–12 of Algorithm 2 is to make each feature $\boldsymbol{d}$ in D perform function $\text{BLOCK}(\boldsymbol{d}, \zeta_i, \zeta_j)$ to generate a new set of subfeatures.

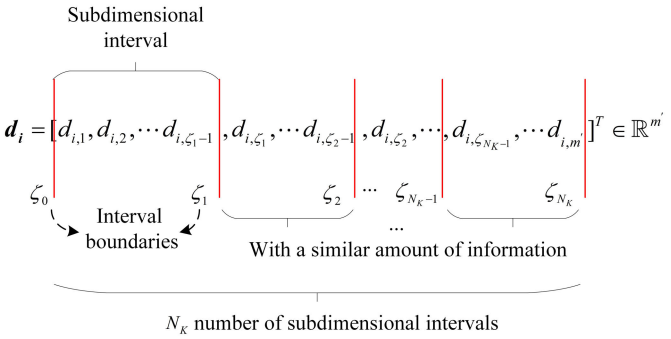


Fig. 3. Subdimensional interval dividing in ASE. N subdimensional intervals are divided, making each interval contain a similar amount of information.

Line 13 in Algorithm 2 is to compute the information amount in each dimension of D^t. Specifically, the eigenvalues of the covariance matrix of $D^{\text{tr}'}$ are used to evaluate the dispersion of orthogonalized features in the corresponding dimension. In theory, the larger the eigenvalue is, the more dispersed the distribution of features along the corresponding dimension is and the stronger the distinguishability of features along this dimension is. Therefore, in the training phase, the percentage of each eigenvalue corresponding to a dimension relative to the sum of all eigenvalues is utilized for evaluating the amount of information along this dimension, which is calculated by

$$\chi_k = \lambda_k \Big/ \sum_{k=1}^{m'} \lambda_k, \quad k = 1, 2, \ldots, m' \tag{6}$$

where k represents the kth dimension. Lines 14–21 in Algorithm 2 are used to divide each feature into N number of subdimensional intervals. Specifically, in order to make contributions of different subdimensional intervals keep similar, the information in different subdimensional intervals needs to be approximately equal to reduce the redundancy of results. Therefore, a set of subdimensional interval boundaries $\mathcal{Z} = \{\zeta_i \in \{1, 2, \ldots, m'\}, i = 0, 1, \ldots, N\}$ is obtained, where ζ_i satisfies

$$\begin{cases} \zeta_i = 1, & i = 0 \\ \zeta_i = \arg\min\limits_{\zeta} \zeta, \quad \text{s.t.} \sum\limits_{k=0}^{\zeta_i} \chi_k \geq \dfrac{i}{N} \quad \text{and} & \\ \qquad \zeta_i \in \{2, 3, \ldots, m'\}, & i > 0. \end{cases} \tag{7}$$

Finally, the preprocessed training feature set D^t is divided into N subfeature sets according to subdimensional interval boundaries in $\mathcal{Z}$, denoted by $\Omega_{D^t} = \{D_i^{t'}, i = 1, 2, \ldots, N\}$, where $D_i^{t'} = \text{DIVIDE}(D^t, \zeta_{i-1}, \zeta_i)$.

c) SD-GMM fitting: After subdimensional interval dividing, subfeatures in each subdimensional interval need to be modeled, to calculate the weights to reflect to distribution characteristics of subfeatures. Considering diversity of the distribution, GMM is selected to model subfeatures in each

Algorithm 2 Subdimensional Interval Dividing

Input: the training feature set D^t after preprocessing, the dimension m' of features, the eigenvalues $\Lambda = diag\{\lambda_1, \lambda_2, \cdots, \lambda_m\}$ obtained by the PCA, the number N of subdimensional intervals.

Output: a set of information amounts $\mathcal{X} = \{\chi_k, k = 1, 2, \cdots, m'\}$, subdimensional interval boundaries $\mathcal{Z} = \{\zeta_i, i = 0, 1, \cdots, N\}$ and the divided sub-feature sets $\Omega_{D^t} = \{D_i^{t\prime}, i = 1, 2, \cdots, N\}$.

function BLOCK($\boldsymbol{d}, \zeta_i, \zeta_j$)
  $\boldsymbol{d^s} = [d_{\zeta_i}, d_{\zeta_i+1}, \cdots, d_{\zeta_j-1}]^T$ where d_k is k-th element in $\boldsymbol{d}$.
  return $\boldsymbol{d^s}$
end function
function DIVIDE(D, ζ_i, ζ_j)
  $D^s = \emptyset$
  for $\boldsymbol{d} = [d_1, d_2, \cdots, d_{m'}]^T$ in D **do**
    $\boldsymbol{d^s} = \text{BLOCK}(\boldsymbol{d}, \zeta_i, \zeta_j)$
    $D^s = D^s \cup \{\boldsymbol{d^s}\}$
  end for
  return D^s
end function
$\lambda^s = \sum_{k=1}^{m'} \lambda_k$
$i = 1, \quad \zeta_0 = 0, \quad \gamma = 0$
for $k = 1 : m'$ **do**
  $\chi_k = \lambda_k / \lambda^s, \quad \gamma = \gamma + \chi_k$
  if $\gamma \geq i/N$ **then**
    $\zeta_i = k$
    $i = i + 1$
  end if
end for
for $i = 1 : N$ **do**
  $D_i^{t\prime} = \text{DIVIDE}(D^t, \zeta_{i-1}, \zeta_i)$
end for

subdimensional interval, and the resultant model is called SD-GMM. For each subfeature set $D_i^{t\prime}$ in Ω_{D^t}, subfeatures in $D_i^{t\prime}$ can be represented by an SD-GMM $\mathcal{G}_i$, which is composed of K_i components. The jth component in $\mathcal{G}_i$ contains three parameters $\alpha_{i,j}$, $\boldsymbol{\mu_{i,j}}$, and $\boldsymbol{\Sigma_{i,j}}$. The probability density of GMM is calculated by

$$p(\boldsymbol{x}) = \sum_{j=0}^{K_i} \alpha_j \cdot \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_j}, \boldsymbol{\Sigma_j}) \tag{8}$$

where

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_j}, \boldsymbol{\Sigma_j}) = \frac{1}{(2\pi)^{\frac{n}{2}} \cdot \left|\boldsymbol{\Sigma_j}\right|^{\frac{1}{2}}} e^{-\frac{1}{2}\cdot[(\boldsymbol{x}-\boldsymbol{\mu_j})^T \boldsymbol{\Sigma_j}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_j})]}. \tag{9}$$

In (8), K_i is computed with the maximum of Bayesian information criterion (BIC) in the D^t, denoted by

$$K_i = \arg\min_K \left(K \cdot \ln(n_t) - 2 \cdot \sum_{j=0}^{n_t} \ln\left(p\left(\boldsymbol{d_j^t}\right)\right) \right)$$
$$\text{s.t. } K_i \in \left\{ 2, 3, \ldots, \frac{N_{\max}}{\left(\prod_{j=1}^{i} K_j \cdot 2^{N-i}\right)} \right\} \tag{10}$$

where $N_{\max}$ is a manually set parameter, representing the desired maximum length of the aggregated vector. After K_i is computed for each subdimensional interval, N numbers of GMMs are fit using the expectation–maximization (E–M) algorithm. It is worth pointing out that, even if multiple GMMs need to be fit, fitting low-dimensional subfeatures only requires a few iterations to converge for each SD-GMM fitting. In addition, the small number of components in each SD-GMM further contributes to a more efficient model fitting.

d) ASE tree construction: In ASE, in order to improve the efficiency, the fitted SD-GMMs are organized by a tree structure called the ASE tree for aggregating features, as shown in Fig. 4. The ASE tree has $N + 1$ levels. The root node is defined by $\mathcal{T}_{0,0}$, and the ith ($i > 0$) level has $\prod_{i=1}^{N} K_i$ nodes. The child nodes of the root nodes correspond to the SD-GMM of the first subdimensional interval, and each child node corresponds to a component of this SD-GMM. Similarly, the set of child nodes of any node in the ASE tree corresponds to an SD-GMM, and nodes in the ith level have their corresponding components in the ith SD-GMM. The jth node $\mathcal{T}_{i,j}$ in the ith level has three types of attributes, i.e., the aggregated element $\kappa_{i,j}$, the weight $\iota_{i,j}$, and the parameters $\alpha_{i,j}$, $\boldsymbol{\mu_{i,j}}$, and $\boldsymbol{\Sigma_{i,j}}$. The parameters of the jth node in the ith level are equal to those of the $\lfloor j/K_i \rfloor$th component in $\mathcal{G}_i$, and they are utilized for calculating $\iota_{i,j}$.

Specifically, $\iota_{i,j}$ and $\kappa_{i,j}$ are intermediate results used for feature aggregation, where $\iota_{i,j}$ represents the weight calculated based on the probability density of the corresponding SD-GMM, and $\kappa_{i,j}$ represents the aggregated result. All the $\kappa_{i,j}$ values at the same level of the ASE tree are concatenated to form a feature vector, which represents the distribution of features across the first j subdimensional intervals. The set of feature vectors from different levels are denoted by $\mathcal{F}$. The feature vector in the ith level is generated from the vector in the $(i-1)$th level and contains more information (the information of feature distribution in the ith dimensional interval) than its up-level vectors. Therefore, with i increases, the corresponding vector is with longer length and contains more information. Meanwhile, the length of vector in the last level of the ASE tree can be limited into $N_{\max}$, which is a manually set parameter introduced in Section III-B3.

For the root node, $\kappa_{0,0}$ and $\iota_{0,0}$ are both set to 1 initially. For the other nodes, $\kappa_{i,j}$ and $\iota_{i,j}$ are calculated during the encoding phase, which is introduced in detail in Section III-B6. In addition, each tree node can visit its next brother node and child nodes.

2) Encoding:

a) Preprocessing: Preprocessing is for orthogonalizing and decorrelating features using z-score and PCA. Given a set of input features $D^{ec} = \{\boldsymbol{d_i^{ec}}, i = 1, 2, \ldots, n^{ec}\}$ for the encoding phase, each feature in D^{ec} is denoted by $\boldsymbol{d_i^{ec}} = [d_{i,1}^{ec}, d_{i,2}^{ec}, \ldots, d_{i,m}^{ec}]^T \in \mathbb{R}^m$. First, for each element $d_{i,j}^{ec}$ in $\boldsymbol{d_i^{ec}}$, it is normalized according to (5). As a result, the normalized feature set $D^{ec'}$ is obtained. Then, $D^{ec'}$ is orthogonalized by each feature in $D^{ec'}$ performing $\boldsymbol{d^e} = \boldsymbol{Q}^{T'} \cdot \boldsymbol{d^{ec'}}$. Finally, the preprocessed feature set $D^e = \{\boldsymbol{d_i^e}, i = 1, 2, \ldots, n^{ec}\}$ is yielded, and each feature in D^e is denoted by $\boldsymbol{d_i^e} = [d_{i,1}^e, d_{i,2}^e, \ldots, d_{i,m'}^e]^T \in \mathbb{R}^{m'}$.

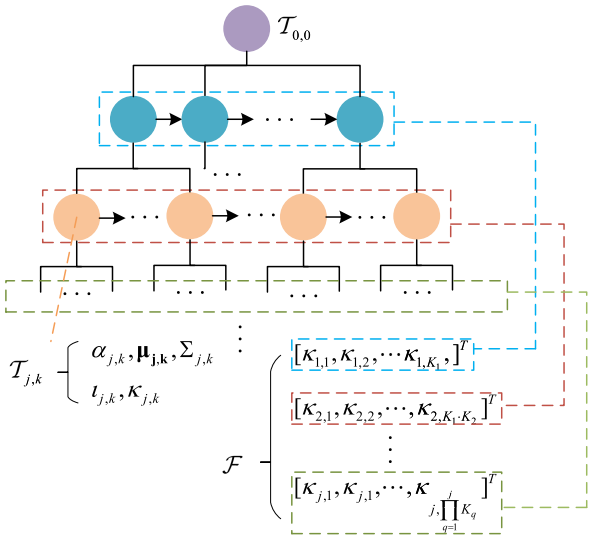


Fig. 4. Structure of the ASE tree with each tree node containing five attributes. $\alpha_{i,j}$, $\boldsymbol{\mu}_{i,j}$, and $\boldsymbol{\Sigma}_{i,j}$ represent three parameters corresponding to a component in the ith SD-GMM, utilized to calculate $\iota_{i,j}$. $\iota_{i,j}$ and $\kappa_{i,j}$ are intermediate results for feature aggregation, calculated using (11) and (12) in the encoding phase, respectively. After encoding, the $\kappa_{i,j}$ values at the same level of the ASE tree are concatenated into a feature vector, representing the distribution of features across the first j subdimensional intervals.

b) Encoding using ASE tree: In the encoding phase of ASE, the constructed ASE tree is used to calculate the weights of each input feature in each subdimensional interval, which is summarized in Algorithm 3. For each feature $\boldsymbol{d}_i^e$ in D^e, it is first segmented into a set of subfeature according to the set of subdimensional interval boundaries $\mathcal{Z}$, denoted by $D_i^s = \{\boldsymbol{d}_{i,j}^s = \text{BLOCK}(\boldsymbol{d}_i^e, \zeta_{j-1}, \zeta_j), j = 1, 2, \ldots, N\}$ and $\boldsymbol{d}_{i,j}^s = [d_{i,j,1}^s, d_{i,j,2}^s, \ldots, d_{i,j,\zeta_j-\zeta_{j-1}}^s]^T$, where $d_{i,j,k}^s = d_{i,\zeta_{j-1}+k}^e$. Next, D_i^s is fed into the ASE tree for performing weight calculating. For instance, $\boldsymbol{d}_{i,j}^s$ is fed into the kth node in the jth level to calculate its weight as follows:

$$\iota_{j,k}^i = \iota_{j-1,\lfloor k/K_j \rfloor}^i \cdot \frac{\alpha_{j,k} \cdot \mathcal{N}\left(\boldsymbol{d}_{i,j}^s | \boldsymbol{\mu}_{j,k}, \boldsymbol{\Sigma}_{j,k}\right)}{\sum_{q=1}^{K_i} \alpha_{j,q} \cdot \mathcal{N}\left(\boldsymbol{d}_{i,j}^s | \boldsymbol{\mu}_{j,q}, \boldsymbol{\Sigma}_{j,q}\right)} \tag{11}$$

where $\iota_{j-1,\lfloor k/K_j \rfloor}^i$ represents the weight of the parent node. When all features in D^e are fed into the ASE tree, each element of the aggregated vector is accumulated by

$$\kappa_{j,k} = \sum_{i=0}^{n^{ec}} \iota_{j,k}^i. \tag{12}$$

Finally, all $\kappa_{i,j}$ values in the jth ($j > 0$) level of the ASE tree are concatenated into a feature vector, which is generated by combining the weights calculated according to the distribution of features in the first j subdimensional intervals. It is denoted by $\boldsymbol{f_j} = [f_1, f_2, \ldots, f_{\prod_{q=1}^{j} K_q}]^T$, where $f_k = \kappa_{j,k} / (\sum_{q=1}^{|f_j|} \kappa_{j,q}^2)^{1/2}$. The vectors obtained from different levels in the ASE tree constitute a set of feature vectors denoted by $\mathcal{F} = \{\boldsymbol{f_j}, j = 1, 2, \ldots, N\}$. As j increases, the longer the length $\boldsymbol{f_j}$ has, the more the information subdimensional intervals $\boldsymbol{f_j}$ contains.

In fact, the way to combine the weights of ASE is the same as that of SE, except that ASE organizes results with a tree structure, which brings three advantages. First, the ASE tree can utilize its tree structure to generate a set of feature vectors with different lengths and different amounts of information, which correspond to different levels in the ASE tree. The feature vector in the jth level is generated from the vector in the $(j-1)$th level and contains overall information of the $(j-1)$th level, as well as the new information in the jth level. Therefore, the feature vectors have different description capabilities, which can be used to accelerate the search phase in PR.

Second, since weights need to be calculated for a feature in different subdimensional intervals, as the number of features in D^e gets larger, the encoding process may be less efficient. To solve the issue, the ASE tree performs dynamic pruning in the encoding process, which greatly improves the efficiency. Specifically, the weight $\iota_{k,j}$ of the node $\mathcal{T}_{k,j}$ is in the range of $[0, 1]$. When $\iota_{k,j}$ is close to 0, the weights of its child nodes are also close to 0, since they must multiply $\iota_{k,j}$. That means contributions of $\mathcal{T}_{k,j}$ and its child nodes to the result are extremely low. Therefore, a pruning threshold δ_p is used. When $\iota_{k,j} < \delta_p$, the calculation processes of $\mathcal{T}_{k,j}$ and its child nodes are ignored, and $\iota_{k,j}$ is set to 0. In the experiments, δ_p is set to 10^{-5}, which ensures the distinguishability of results and improves the efficiency.

Third, with the tree structure, ASE does not need to calculate indices of aggregated vectors, which can lead to a more efficient encoding phase.

IV. ASE-Based PR

A. Problem Definition

For a place $\mathcal{P}$, sensor data $\mathcal{D}_\mathcal{P}$ are obtained using the onboard sensors of the robot, e.g., cameras or LiDARs. Then, a set of local descriptors $\mathcal{L}_\mathcal{P} = \{\boldsymbol{l_i}, i = 1, 2, \ldots, n_i\}$ is extracted from $\mathcal{D}_\mathcal{P}$, which represents appearance characteristics of $\mathcal{P}$. For the current place $\mathcal{P}_c$, if a set of similar local descriptors is retrieved from the historical data, the robot is considered to have visited $\mathcal{P}_c$ previously. In order to retrieve the similar set of local descriptors more efficiently, these local descriptors are inputted into the ASE method and aggregated into a set of feature vectors, each of which is a global descriptor to represent $\mathcal{P}_c$ with low dimensions. As a result, the set of global descriptors is obtained, denoted by $\mathfrak{G}_\mathcal{P} = \text{ASE}(\mathcal{L}_\mathcal{P})$, where $\mathfrak{G}_\mathcal{P} = \{\boldsymbol{g}_\mathcal{P}^i, i = 1, 2, \ldots, N\}$ and $\boldsymbol{g}_\mathcal{P}^i$ is the feature vector in the ith level of the ASE tree. So far, PR is converted into the problem of retrieving the most similar set of global descriptors $\mathfrak{G}_{\mathcal{P}_j}$ to $\mathfrak{G}_{\mathcal{P}_c}$ from historical sets of global descriptors. If the similarity between $\mathfrak{G}_{\mathcal{P}_j}$ and $\mathfrak{G}_{\mathcal{P}_c}$ is high enough, $\mathcal{P}_c$ and $\mathcal{P}_j$ are considered to be the same place. Otherwise, $\mathcal{P}_c$ is considered to be a new place which the robot has not visited before.

As introduced in Section III-B, in ASE, although a simple training process is required to divide subdimensional intervals and fit SD-GMMs, the ASE-based framework can meet the requirements of online training in PR due to the fast training process. Hence, for the ASE-based PR framework, the first

Algorithm 3 Encoding Process of ASE

Input: the feature set D^e after preprocessing, the dimension m' of features, the root node $\mathcal{T}_{0,0}$ of the ASE tree, the pruning threshold δ_p

Output: the set of aggregated feature vectors $\mathcal{F} = \{\boldsymbol{f_j}, j = 1, 2, \cdots, N\}$

function COMPUTEWEIGHT($\mathcal{T}_{j,k}$, D_i^s)
  if PARENT($\mathcal{T}_{j,k}$) exists **then**
    Find the j-th element $d_{i,j}^s$ from D_i^s
    Compute the weights $\iota_{j,k}$ of $\mathcal{T}_{j,k}$ using $d_{i,j}^s$ by (11)
    if $\iota_{j,k} < \delta_p$ **then**
      $\iota_{j,k} = 0$
    end if
  end if
  if $\iota_{j,k} \neq 0$ **then**
    $\Omega_{\mathcal{T}} =$ CHILDREN($\mathcal{T}_{j,k}$)
    for $\mathcal{T}$ in $\Omega_{\mathcal{T}}$ **do**
      COMPUTEWEIGHT($\mathcal{T}$, D_i^s)
    end for
  end if
end function
for $\boldsymbol{d_i^e}$ in D^e **do**
  $D_i^s = \{\boldsymbol{d_{i,j}^s} =$ BLOCK($\boldsymbol{d_i^e}$, ζ_{j-1}, ζ_j), $j = 1, 2, \cdots, N\}$
  COMPUTEWEIGHT($\mathcal{T}_{0,0}$, D_i^s)
  for $j = 1 : N$ **do**
    for $k = 1 : \prod_{q=1}^{j} K_q$ **do**
      Compute the $\kappa_{j,k}$ in the feature vector by (12)
      $f_k = \kappa_{j,k}$
    end for
  end for
  $\boldsymbol{f_j} = [f_1, f_2, \cdots, f_{\prod_{q=1}^{j} K_q}]^T$
  Calculate L-2 normalization for $\boldsymbol{f_j}$
end for

K_d frames of data acquired by sensors are utilized for online training of ASE. After training, the proposed framework can utilize the constructed ASE tree to generate a set of feature vectors as global descriptors and perform PR using a hierarchical matching strategy.

B. Hierarchical Matching

As more places are visited, the computational time for matching the current global descriptors to historical ones becomes longer. The ASE tree structure describes a place using multiple global descriptors, which vary in the length and description capability. Benefiting from the high distinguishability of the ASE-generated descriptors, in many instances, historical places exhibiting significant appearance differences with the current place can be efficiently filtered out by performing a shorter length global descriptor matching. Therefore, a hierarchical matching strategy is designed to improve the efficiency of PR significantly without sacrificing precision.

For a place $\mathcal{P}$, the set of global descriptors $\eth_{\mathcal{P}} = \{\boldsymbol{g_{\mathcal{P}}^i}, i = 1, 2, \ldots, N\}$ ($|\boldsymbol{g_{\mathcal{P}}^i}| = \prod_{j=1}^{i} K_i$) aggregated by ASE is available for matching with the historical set of descriptors. The Euclidean distances between the global descriptors in the same level are utilized to evaluate the similarity of two places. The smaller the distance between $\boldsymbol{g_{\mathcal{P}_k}^i}$ and $\boldsymbol{g_{\mathcal{P}_j}^i}$ is, the more likely place $\mathcal{P}_k$ is the same with $\mathcal{P}_j$. Specifically, the amount of information contained in each global descriptor $\boldsymbol{g_{\mathcal{P}}^i}$ is evaluated by

$$\gamma_i = \sum_{j=1}^{\zeta_i} \chi_j. \tag{13}$$

As i gets larger, $\boldsymbol{g_{\mathcal{P}}^i}$ contains richer information and has longer length.

In the hierarchical matching strategy, most of unmatched places are quickly filtered out by global descriptors with short lengths. In contrast, if global descriptors with longer lengths are used for matching, more accurate recognition results can be obtained. Assuming that N_H historical places have been visited, the proposed strategy utilizes the global descriptor $\boldsymbol{g_{\mathcal{P}}^i}$ obtained in the ith level of the ASE tree to filter out $\lfloor (N_H - 1) \cdot \gamma_i \rfloor$ number of unmatched historical places, where γ_i $(0 < \gamma_i \leq 1)$ represents the information amount to the ith level of the ASE tree. As a result, $N_H - \lfloor (N_H - 1) \cdot \gamma_i \rfloor$ number of matched places is reserved for matching in the next level. At each level of matching, the information amount is employed to determine the number of candidates retained for the next level of matching. This strategy offers greater flexibility and adaptability compared with using a fixed threshold. Finally, the above matching process is performed hierarchically until the potentially identical place to the current place is obtained through matching global descriptors in the last level of the ASE tree.

The hierarchical matching strategy improves the retrieval efficiency in PR. Theoretically, assuming that K_i in each interval is set to a constant K, hierarchical matching is approximately $1/(\sum_{i=1}^{N}((N + 1 - i)/(N \cdot K^{N-i})))$ faster than direct matching. As K gets larger, improvement in matching efficiency is closer to N times higher than direct matching.

V. EXPERIMENTS

In this section, extensive experiments are carried out on the public data sequences KITTI [34] and in real-world scenes. Specifically, the experimental setup is first introduced in Section V-A. Then, ASE is compared with four feature aggregation methods in Section V-B. In Section V-C, the proposed framework for PR is compared with the state-of-the-art PR methods using images and LiDAR data, respectively. The efficiency of the ASE-based PR framework is evaluated in Section V-D. Finally, the fusion experiment of images and LiDAR data is carried out in Section V-E.

A. Experimental Setup

In this article, the KITTI and self-built NKU datasets are utilized to evaluate ASE and the ASE-based PR framework. In the KITTI dataset, five sequences are chosen, including 00, 02, 05, 06, and 07, which contain 4541, 4660, 2671, 1101,

and 1101 frames of images and LiDAR data, respectively. Each image has 1271 × 376 pixels, and each of LiDAR data has 12 000 scan points approximately, of which the maximal range is about 100.0 m. The images and LiDAR data in KITTI are collected by a stereo camera and a Velodyne HDL-64H LiDAR installed on a car moving on a street. In addition, the images and LiDAR data are preprocessed synchronously. In this article, only the left camera is utilized in the experiments. For the real-scene dataset NKU, 311 frames of images and LiDAR data are collected by a monocular camera and a Velodyne HDL-32 LiDAR installed on a Pioneer3 DX mobile robot moving in an indoor corridor. Each image has 1920 × 1080 pixels, and each of LiDAR data has 60 000 scan points approximately, of which the maximal range is about 40.0 m.

In the experiments, for the KITTI dataset, two frames of data are considered to be collected from the same place if their distance is less than 6.0 m and different places if their distance is more than 6.0 m, such as [30]. For the NKU dataset, the ground truth is marked manually. In order to avoid repeated recognition of the same place, adjacent 50 frames of data are excluded from the retrieval scope. Under this situation, there are 817, 266, 513, 271, 86, and 184 positive samples in KITTI 00, 02, 05, 06, 07, and NKU sequences, respectively. All the experiments are carried out on a computing platform with Intel i7-8700 2.6-GHz CPU, 8-GB memory, and Ubuntu 18.04 operating system. All the experiments do not utilize GPU acceleration. In addition, the open source code of ASE is released online,[1] and the self-built dataset NKU is publicly published.[2]

B. Evaluation on ASE

In this section, ASE is compared with four unsupervised feature aggregation methods, which are widely used in PR, including BoW [12], incremental BoW (IBoW) [30], VLAD [13], and SE [11]. In the experiments, diverse methods are employed to extract local descriptors from data obtained from different sensors. These local descriptors are subsequently aggregated into global descriptors. As new sensor data are captured, ASE is continuously utilized to generate global descriptors, which are then compared against historical global descriptors to determine if the current place matches any previous one. Finally, by adjusting the matching threshold, multiple pairs of precision and recall values are obtained to plot the precision–recall curve. For BoW, K-means is utilized to cluster features into multiple clusters. Then, features in each cluster are counted for generating feature vectors. IBoW incrementally reclusters small amounts of features when the number of features in one BoW tree node is larger than the preset thresholds. VLAD expands BoW by adding the distance of each feature from its cluster center. SE analyzes the distribution of features in different dimensions and calculates weights for measuring to what extent a feature belongs to the corresponding bin. In order to evaluate the distinguishability of aggregated results, the same set of local descriptors is aggregated by different feature aggregation methods to generate global descriptors. Then, PR is performed by matching the resultant global descriptors. Precision–recall performances are adopted to quantitatively evaluate the distinguishability of these methods.

Since BoW, VLAD and ASE need to be trained, in this experiment, the first 10% data in the sequences are selected for training. Then, the remaining data are utilized for feature aggregation and matching. In contrast, IBoW and SE do not require a training phase. Thus, it directly performs feature aggregation and matching. KITTI 05, 06, and 07 are selected in the experiments. For each sequence, ORBs and SIFTs are extracted from images, while 3-D SUs and SRDs are extracted from LiDAR data, respectively. In general, as the length of the aggregated feature vectors gets longer, the feature vectors are more informative and distinguishable, and matching them with other feature vectors takes more computational resources. Therefore, in order to ensure fairness of the experimental comparison, lengths of aggregated vectors in four methods need to be approximately equal to each other. For this reason, some parameters are set to ensure the feature vectors aggregated by each method have a maximum length, denoted by $L_{\max}$, which is set to 10 000 in our experiments. For instance, the clustering number of BoW and VLAD and the maximum clustering number of IBoW are set to $L_{\max}$, $\lfloor L_{\max}/m \rfloor$, and $L_{\max}$, respectively, where m is the dimension of features. The number of bins in SE is set to $\lfloor L_{\max}^{(1/m)} \rfloor - 1$. For ASE, $N_{\max}$ is set to $L_{\max}$, and N is set to 7. Fig. 5 shows the precision–recall curve of PR using the global descriptors aggregated by different methods with different KITTI sequences.

Compared with BoW, IBoW, and VLAD, ASE obtains better results for aggregating different local descriptors extracted from different types of sensor data in almost all sequences. Only in KITTI 06 sequence, ASE performs a little worse than VLAD for aggregating SRDs. Compared with the clustering-based methods, although ASE employs the PCA for dimension reduction, more than 95% of information is retained in features while discarding the information susceptible to noises. In addition, ASE utilizes multiple SD-GMMs to fit the distribution of features in subdimensional intervals and combines weights from different subdimensional intervals to generate vectors, which retains more information than clustering-based methods. Compared with SE, ASE can aggregate features with much higher dimensions. For instance, when aggregating ORBs with 32 dimensions, SIFTs with 128 dimensions, and SRDs with 125 dimensions, the weights calculated by SE in every single dimension are combined, which makes the lengths of aggregated vectors (greater than 2^{32}, 2^{128}, and 2^{125}) exceed the storage capabilities of our computing devices. In contrast, ASE can limit the length of the feature vector to $N_{\max}$ and obtain good results for all types of features, which demonstrates the superiority of ASE over SE.

In addition, the training time and total aggregation time for visual descriptors ORBs and LiDAR descriptors 3-D SUs are evaluated on the KITTI 05 sequence, as shown in Tables I and II, respectively. When the robot visits a

[1][Online]. Available: https://github.com/wdyiwdwd/ASE-Encoder

[2][Online]. Available: https://drive.google.com/file/d/1nzscGa1VHEMyFdDGOFSkaeK_FW5U2w4t/view?usp=sharing

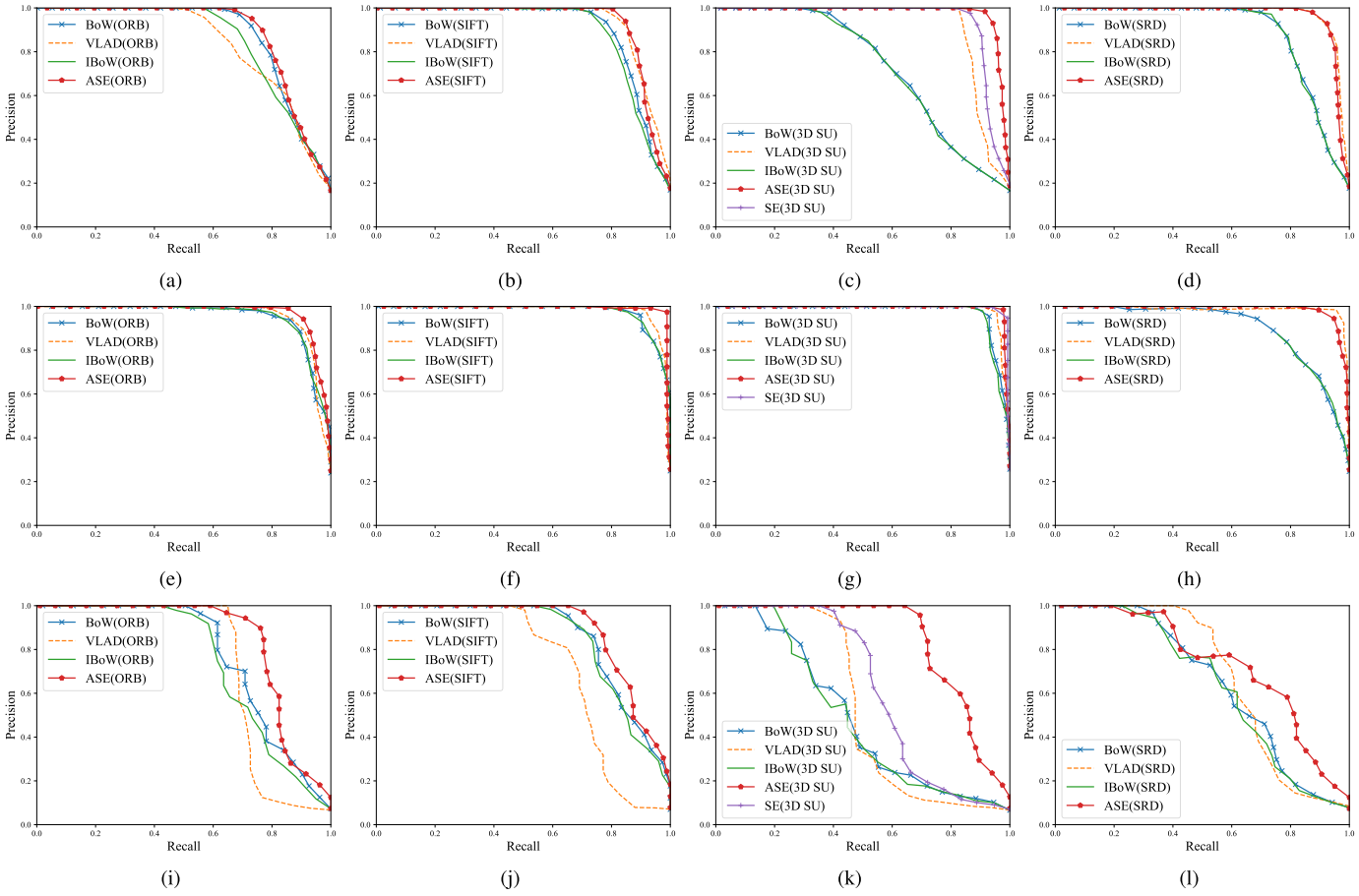


Fig. 5. Precision–recall curves of PR using the global descriptors aggregated by different methods on different sequences. (a) KITTI 05 ORB. (b) KITTI 05 SIFT. (c) KITTI 05 3-D SU. (d) KITTI 05 SRD. (e) KITTI 06 ORB. (f) KITTI 06 SIFT. (g) KITTI 06 3-D SU. (h) KITTI 06 SRD. (i) KITTI 07 ORB. (j) KITTI 07 SIFT. (k) KITTI 07 3-D SU. (l) KITTI 07 SRD.

completely unknown environment, a fast training process is required to efficiently model appearance characteristics of the environment and understand complex situations and further to quickly response to potentially hazardous conditions. For ASE, the most time-consuming step is the SD-GMM fitting, which usually takes more than 90.0% of the total training time. However, owing to PCA and dividing features into multiple subdimensional intervals, ASE only needs to use low-dimensional subfeatures to fit SD-GMM models, where each SD-GMM model has only a small number of components, making this step quickly converge after a few iterations. In addition, due to the independent fitting process of each SD-GMM model, the parallel implementation of ASE is not affected by conflicts caused by multithreaded, which further improves its efficiency.

For both ORBs and 3-D SUs, the training time of ASE is 7.2305 and 10.1199 s, respectively, which is much smaller than that of BoW and VLAD. That means that ASE can be trained online for applications in unknown environments. Although IBoW and SE do not need to be trained, IBoW has to recluster features in the aggregation process, and thus, its aggregation time is much longer than other methods. Likewise, although SE performs efficiently when aggregating 3-D SUs, it cannot aggregate ORBs with 32 dimensions, which limits its generalizability. In addition, aggregating ORBs with ASE takes a little longer time than BoW. However, the improvement in the precision and recall of ASE results can compensate for the slight increase in aggregation time. Meanwhile, because KITTI 05 contains 2671 frames of data, the average aggregation time for ORBs and 3-D SUs in KITTI 05 are only 0.0378 and 0.0086 s, respectively, which ensures that ASE is well suited for online PR with images or LiDAR data.

TABLE I

TRAINING AND TOTAL AGGREGATION TIME (S) OF DIFFERENT METHODS TO AGGREGATE ORBS ON THE KITTI 05 SEQUENCE

	Training time	Total aggregation time	Totals
BoW	474.9959	**67.3893**	542.3852
IBoW	**0.0**	362.1420	362.1420
VLAD	45.9842	72.1170	118.1012
SE	-	-	-
ASE	7.2305	101.0894	**108.3199**

It is worth pointing out that ASE not only has adaptability to different types and dimensions of input features, but also can generate an effective model with only a small amount of training data. The impact of the amount of training data on the performances of three methods with training is evaluated when aggregating ORBs and 3-D SUs in KITTI 05, as shown

TABLE II
TRAINING AND AGGREGATION TIME (S) OF DIFFERENT METHODS TO AGGREGATE 3-D SUS ON THE KITTI 05 SEQUENCE

	Training time	Total aggregation time	Totals
BoW	198.5705	76.8180	275.3885
IBoW	**0.0**	192.3412	192.3412
VLAD	32.0219	83.9228	115.9447
SE	**0.0**	**13.4620**	**13.4620**
ASE	10.1199	22.9706	33.0905

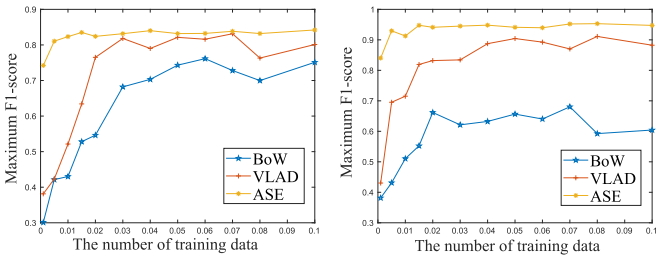


Fig. 6. Impact of the amount of training data on the performances of three unsupervised methods for aggregating (a) ORBs and (b) 3-D SUs, in which the x-axis represents the percentage of the training data to all data, while the y-axis represents the maximum $F1$-score of PR using the aggregating vectors.

in Fig. 6. ASE achieves relatively stable results when only the first 1.5% of the data are used for training. In contrast, VLAD and BoW require more than 4.0% of the data for training to obtain stable results. For different amounts of training data, performances of ASE feature vectors are more stable than those of the other two methods. This is because VLAD and BoW rely heavily on clustering results. If similar features are clustered into different clusters, they would yield a serious impact on results. In comparison, ASE uses SD-GMMs to calculate probability densities as weights, which ensures that the weights calculated by the similar features are approximately equal to each other. Therefore, ASE can generate a more stable environment representation model in various scenes.

C. PR Evaluation

Unlike the experiments in Section V-B, where only the feature aggregation methods are different, in this section, the proposed framework is compared with six state-of-the-art PR methods in KITTI 00, 02, 05, 06, 07, and NKU sequences using precision–recall curves, such as [23], [24], [27], [30], of which GIST [20], FAB-MAP [14], and the image-sequence-based PR (ISB-PR) [27] use images, while the SHOT [18], scan context (SC) [24], and SRG-PR [30] use LiDAR data. They include both local descriptor-based methods (ISB-PR, FAB-MAP, SHOT, and SRG-PR) and global descriptor-based methods (GIST and SC).

GIST is a global descriptor extracted from images for describing the spatial structure of the environment. For FAB-MAP, SURFs are extracted from images and then aggregated into a binary global descriptor by a probabilistic BoW approach. For ISB-PR, PIRFs are extracted from image sequences, and BoW is utilized for matching the current and historical data. SHOT is a LiDAR local descriptor based on the histogram, which counts the normal vectors in the normalized spherical area. SC is a global descriptor extracted from LiDAR data, which counts the highest points in the fan-shaped area on the $x\,y$ plane. SRG-PR segments LiDAR data and constructs an SRG to represent the environment; then, PR is achieved with SRG matching. In implementation of GIST, FAB-MAP, and SC, we use the open-source code released by the authors, and parameters are set as the default values. For SHOT, a C++ version implemented in the point cloud library (PCL)[3] is used. For the proposed framework, the parameters of ASE are set as the same values as those in Section V-B.

For PR with images or LiDAR data, precision–recall curves of different methods are shown in Fig. 7. Meanwhile, maximum $F1$-scores of each method using images and LiDAR data are listed in Tables III and IV, respectively. GIST is limited by the descriptive ability of the structural characteristics, and thus, it does not obtain the best performance in any sequence. FAB-MAP is limited by the distinguishability of vectors aggregated by BoW, making the recognition results worse than those of the proposed framework in all sequences. ISB-PR performs well in all sequences and obtains best results in KITTI 06 and NKU, because ISB-PR utilizes an ESN to model image sequences, which contains much richer information than a single image. However, ISB-PR is affected by the difference in the angle of view in a part of image sequences, which results in more missed detection. In comparison, the proposed framework benefits from the strong distinguishability of feature vectors aggregated by ASE, such that the best performance can be achieved in KITTI 00, 02, 05, and 07 when aggregating SIFTs and ORBs, which demonstrates the effectiveness of the proposed framework. In addition, in all sequences, the ASE-based framework can achieve good recalls at 100% precision. For instance, although the maximum $F1$-scores of the ASE-based method are a little lower than those of ISB-PR in the NKU sequence, the recalls at 100% precision are 15.89% and 9.37% better than those of ISB-PR when aggregating ORBs and SIFTs, respectively.

For the results using LiDAR data, SHOT does not obtain good performances in all sequences, because it is not robust to subtle changes caused by noises. Performances of SC are good in KITTI 00, 02, 05, and 06. However, SC only counts the highest point in each area. As a result, distinguishing structured environments in NKU is difficult for SC. SRG-PR obtains the best performance in KITTI 06 and 07. However, it relies heavily on the segmentation results of point clouds. For instance, in KITTI 02, there are lots of bushes in the environment, which causes difficulty in point cloud segmentation. As a result, SRG-PR obtains poor performances. In contrast, the proposed framework for aggregating 3-D SUs performs well in each LiDAR sequence. It obtains the best performance in KITTI 00, 02, 05, and NKU sequences. Meanwhile, the proposed framework for aggregating SRDs also yields good results in KITTI 00, 05, 06, 07, and NKU. It is worth pointing out that the proposed framework for

[3][Online]. Available: http://pointclouds.org

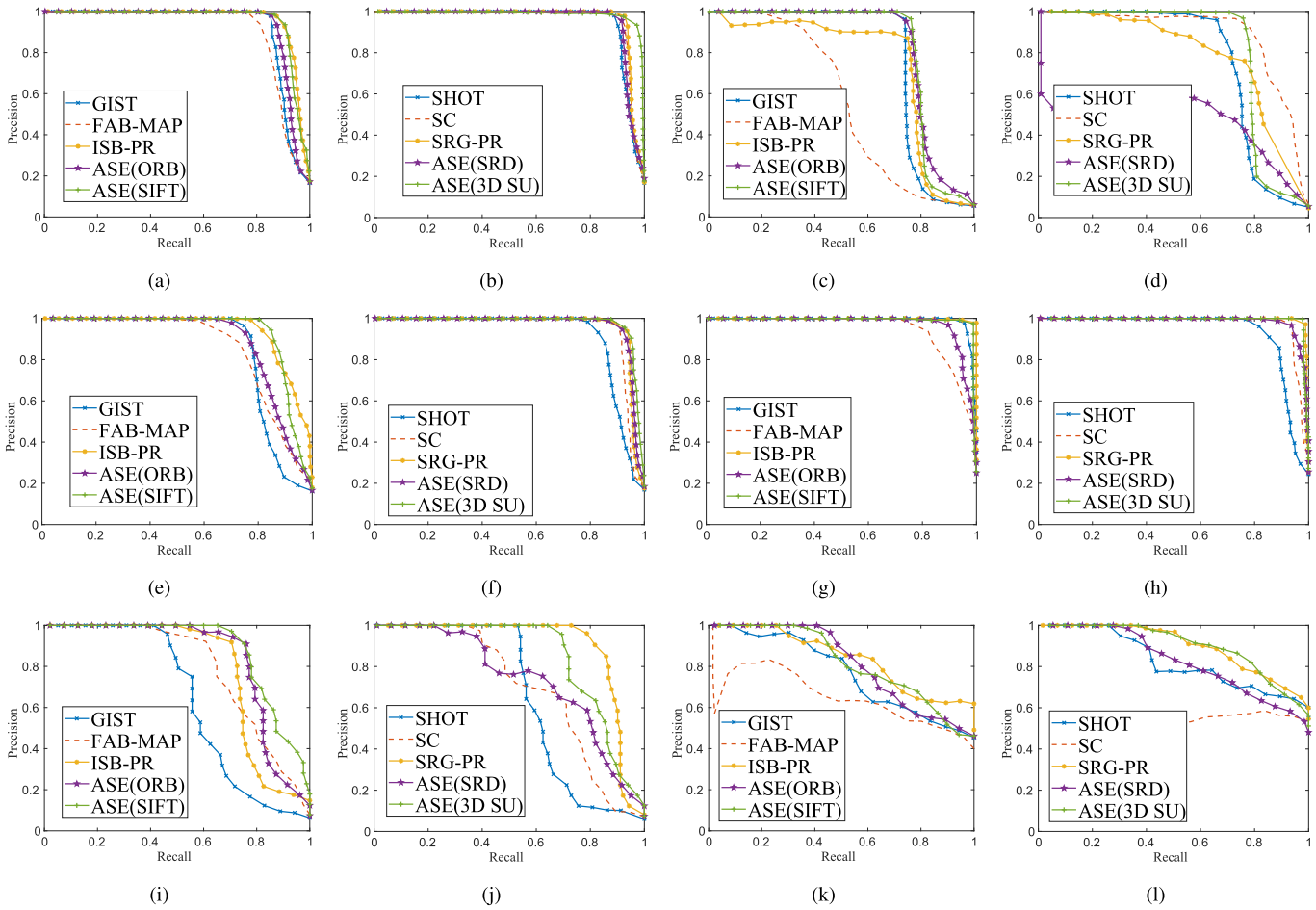


Fig. 7. Precision–recall curve of each method for PR with images or LiDAR data. (a) KITTI 00 using images. (b) KITTI 00 using LiDAR data. (c) KITTI 02 using images. (d) KITTI 02 using LiDAR data. (e) KITTI 05 using images. (f) KITTI 05 using LiDAR data. (g) KITTI 06 using images. (h) KITTI 06 using LiDAR data. (i) KITTI 07 using images. (j) KITTI 07 using LiDAR data. (k) NKU using images. (l) NKU using LiDAR data.

TABLE III
MAXIMUM $F1$-SCORES OF EACH METHOD FOR PR WITH IMAGES

	GIST	FAB-MAP	ISB-PR	ASE (ORB)	ASE (SIFT)
KITTI 00	0.9114	0.8774	0.9273	0.9156	**0.9333**
KITTI 02	0.8436	0.5781	0.8133	0.8380	**0.8526**
KITTI 05	0.8485	0.7996	0.8741	0.8319	**0.9006**
KITTI 06	0.9716	0.8752	**0.9962**	0.9338	0.9870
KITTI 07	0.6420	0.7362	0.7976	**0.8284**	0.8235
NKU	0.6667	0.6536	**0.7665**	0.6807	0.7279

TABLE IV
MAXIMUM $F1$-SCORES OF EACH METHOD FOR PR WITH LIDAR DATA

	SHOT	SC	SRG-PR	ASE (SRD)	ASE (3D SU)
KITTI 00	0.9332	0.9459	0.9543	0.9412	**0.9625**
KITTI 02	0.7841	0.8500	0.7623	0.5895	**0.8515**
KITTI 05	0.8856	0.9408	0.9481	0.9360	**0.9482**
KITTI 06	0.8945	0.9655	**0.9907**	0.9533	0.9888
KITTI 07	0.6939	0.6769	**0.8670**	0.6936	0.8098
NKU	0.7772	0.7095	0.8000	0.7250	**0.8036**

aggregating SRDs performs a little worse than SRG-PR, because in SRG-PR, correspondences are established for SRDs by precise but time-consuming graph matching, while in the proposed framework, only the distribution of SRDs is fit. Although no one-by-one matching of local descriptors is used, our framework still obtains good results with less computing resources.

Fig. 8 shows the visualization results of the ASE-based framework at $R_{P_{100}}$ (i.e., maximum recall at 100% precision) when aggregating SIFTs and 3-D SUs in KITTI sequences, respectively, where true positives are marked by red and false negatives by black, such as [26]. In all the sequences, the proposed framework recalls most of positive samples (red trajectory) along the revisited trajectory (red and black trajectories), demonstrating the ASE-based framework performs well in the condition without false alarms. Therefore, the proposed framework can be also used for loop closure detection in a SLAM system.

D. Evaluation on Efficiency

In this section, the average computational time of different methods is evaluated, including average description time and average search time. As the sensor data are acquired, the PR methods describe and represent the data and stored the descriptive information. After describing the data of the current frame, similar data are retrieved from the historical

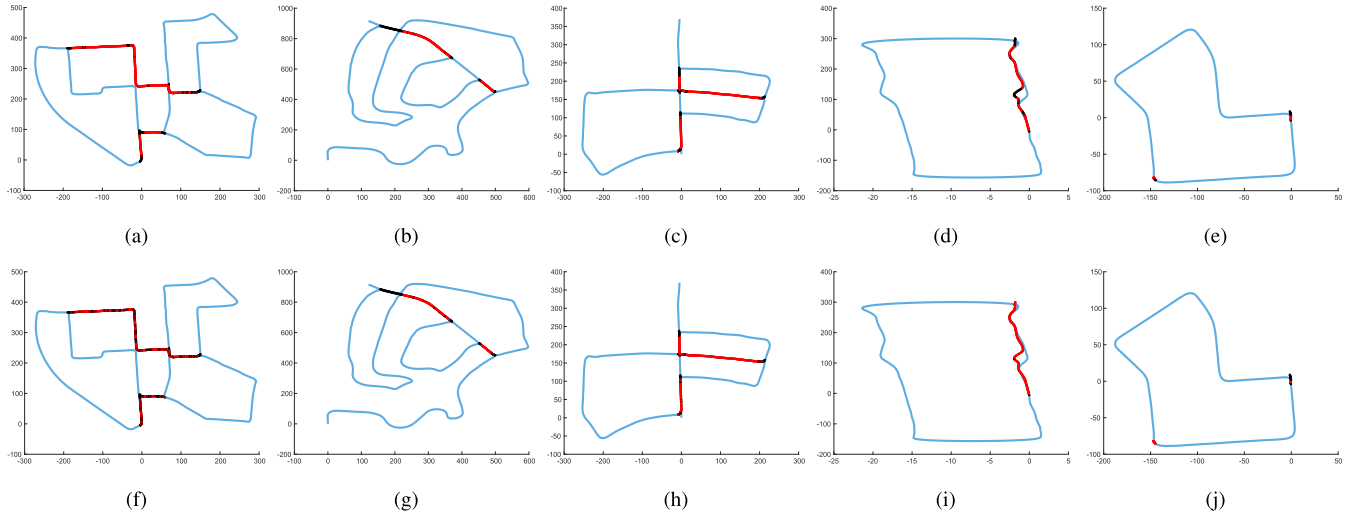


Fig. 8. Visualization results of the proposed framework at R_{P100} when aggregating SIFTs and 3-D SUs, where true positives are marked by red and false negatives by black. The proportion of the red trajectory to the total red and black trajectories represents the maximum recall at 100% precision. (a) KITTI 00 SIFT. (b) KITTI 02 SIFT. (c) KITTI 05 SIFT. (d) KITTI 06 SIFT. (e) KITTI 07 SIFT. (f) KITTI 00 3-D SU. (g) KITTI 02 3-D SU. (h) KITTI 05 3-D SU. (i) KITTI 06 3-D SU. (j) KITTI 07 3-D SU.

TABLE V
AVERAGE COMPUTATIONAL TIME (S) OF DIFFERENT METHODS WITH IMAGES IN KITTI 05

	Description time	Search time	Total time
GIST	0.1619	0.0191	0.1790
FAB-MAP	0.0955	0.0921	0.1876
ISB-PR	0.3874	0.0127	0.4001
ASE (ORB)	**0.0509**	**0.0097**	**0.0606**
ASE (SIFT)	0.0858	0.0113	0.0971

TABLE VI
AVERAGE COMPUTATIONAL TIME (S) OF DIFFERENT METHODS WITH LIDAR DATA IN KITTI 05

	Description time	Search time	Total time
SHOT	0.5589	0.1042	0.6631
SC	0.1386	0.3274	0.4660
SRG-PR	0.2124	0.2355	0.4479
ASE (SRD)	0.1846	0.0132	0.1978
ASE (3D SU)	**0.0881**	**0.0094**	**0.0975**

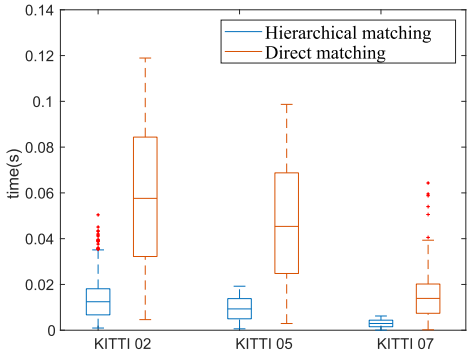


Fig. 9. Box plot of the search time with hierarchical matching and direct matching, respectively. The middle line represents the median of the search time. The upper and lower boundaries of the box depict the upper and lower quartiles, respectively. The upper and lower short lines out of the box represent the maximum and minimum of the search time, respectively. The red symbols depict some outliers caused by increasing the search space in KITTI 02.

data using the descriptive information and then determine whether the robot visits the same place. The description time represents the time consumed to model the sensor data, while the search time represents the time spent on retrieving similar data from the historical records. In our experiment, the average computational time in KITTI 05 sequence for different methods with images and LiDAR data is listed in Tables V and VI, respectively.

The description of the ASE-based PR framework is more efficient using both images and LiDAR data because of the high efficiency of ASE. Meanwhile, hierarchical matching makes the proposed framework more efficient for search, even better than matching the global descriptor GIST with only 1536 dimensions. Therefore, the efficiency of the proposed framework can meet the requirement for online PR even with high frequency of data collection. In addition, it is worth pointing out that the proposed framework takes slightly long time for description using SRDs, because the extraction of SRDs is time-consuming. In fact, the average aggregation time of ASE for SRDs takes only 0.0374 s in KITTI 05.

Finally, performances of hierarchical matching are evaluated in KITTI 02, 05, and 07 sequences. In the experiment, 3-D SUs are aggregated into global descriptors by ASE. The box plot of the search time with hierarchical matching and direct matching is drawn in Fig. 9. Moreover, the corresponding maximum $F1$-scores are listed in Table VII. The results demonstrate that the hierarchical matching strategy does not have an obvious impact on the precision–recall performances of PR. Nevertheless, it can greatly improve the efficiency of the search process, including the average, maximum, and variance of the search time. Moreover, the improvement on efficiency is close to N times (N is set to 7 in the experiments), which is consistent with the theoretical analysis in Section IV-B.

TABLE VII

MAXIMUM $F1$-SCORES WITH OR WITHOUT THE HIERARCHICAL MATCHING STRATEGY IN EACH SEQUENCE

	KITTI 02	KITTI 05	KITTI 07
With the strategy	0.8515	0.9482	0.8098
Without the strategy	0.8604	0.9324	0.8025
Differences	-0.0089	+0.0158	+0.0073

TABLE VIII

MAXIMUM $F1$-SCORES OF PR USING THE FUSED AND INDIVIDUAL DESCRIPTORS ON KITTI 05 AND 06 SEQUENCES

	SIFTs	SRDs	Fusion
KITTI 05	0.9006	0.9360	**0.9439**
KITTI 06	0.9870	0.9533	**0.9926**

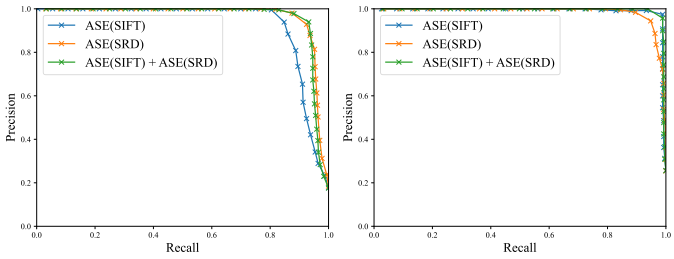


Fig. 10. Precision–recall curves of PR using the fused and individual descriptors on KITTI 05 and 06 sequences. (a) KITTI 05. (b) KITTI 06.

E. Evaluation on Fusion

In this section, a simple fusion experiment is carried out. First, SIFTs and SRDs are extracted from synchronized images and LiDAR data, respectively. Then, ASE is utilized to aggregate them into global descriptors. Next, these two types of descriptors are cascaded into a single global descriptor for environment representation. Finally, PR is performed using the cascaded descriptors on KITTI 05 and 06 sequences. The precision–recall curves are shown in Fig. 10, and the maximum $F1$-scores are listed in Table VIII. The results demonstrate a slight improvement in the fused descriptor, indicating the potential of the ASE-based PR framework for multisensor fusion. To further improve the performance of multisensor fusion, a deeper level of fusion mechanism is required in the future.

VI. CONCLUSION

In this article, ASE was proposed for aggregating a set of features into feature vectors with low dimensions. First, input features have been preprocessed and then fed into the root node of the ASE tree. Next, weights have been calculated by probability densities of SD-GMMs in the ASE tree node to generate aggregated vectors. Meanwhile, an ASE-based PR framework has been proposed, in which local descriptors have been extracted from sensor data (e.g., images or LiDAR data), and then aggregated as global descriptors for representing environment. In addition, a hierarchical matching strategy has been proposed for retrieving similar descriptors to accomplish PR. The proposed framework for PR has been compared with the state-of-the-art methods on the public dataset and in real-world scenes, which demonstrates its good performances on both accuracy and efficiency in various situations.

Compared with SE, ASE calculates the weights according to the distribution of features in subdimensional intervals instead of every single dimension, which makes ASE extensible to any dimension of input features. Moreover, the proposed framework for PR is theoretically independent of the sensor type as well as the specific method of local descriptor extraction. Furthermore, as a generic feature aggregation method, ASE is also suitable for global localization and loop closure detection in SLAM.

In future work, we would like to use ASE to aggregate local descriptors extracted from different types of sensors in a unified framework to accomplish deep fusion of multimodal sensor data, to further improve the accuracy and robustness of PR.

REFERENCES

[1] X. Zhang, L. Wang, and Y. Su, "Visual place recognition: A survey from deep learning perspective," *Pattern Recognit.*, vol. 113, May 2021, Art. no. 107760.

[2] J. Ma, X. Chen, J. Xu, and G. Xiong, "SeqOT: A spatial–temporal transformer network for place recognition using sequential LiDAR data," *IEEE Trans. Ind. Electron.*, vol. 70, no. 8, pp. 8225–8234, Aug. 2023.

[3] T. Ye, X. Yan, S. Wang, Y. Li, and F. Zhou, "An efficient 3-D point cloud place recognition approach based on feature point extraction and transformer," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–9, 2022.

[4] G. He, Q. Zhang, and Y. Zhuang, "Online semantic-assisted topological map building with LiDAR in large-scale outdoor environments: Toward robust place recognition," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–12, 2022.

[5] L.-P. Berczi and T. D. Barfoot, "Looking high and low: Learning place-dependent Gaussian mixture height models for terrain assessment," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 3918–3925.

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

[7] J. Yan, X. Li, X. Yang, X. Luo, C. Hua, and X. Guan, "Integrated localization and tracking for AUV with model uncertainties via scalable sampling-based reinforcement learning approach," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 11, pp. 6952–6967, Nov. 2022.

[8] Q. Sun, J. Yuan, X. Zhang, and F. Duan, "Plane-edge-SLAM: Seamless fusion of planes and edges for SLAM in indoor environments," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 4, pp. 2061–2075, Oct. 2021.

[9] Q. Sun, J. Yuan, and X. Zhang, "IT-HYFAO-VO: Interpretation tree-based VO with hybrid feature association and optimization," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–18, 2021.

[10] J. Yuan, S. Zhu, K. Tang, and Q. Sun, "ORB-TEDM: An RGB-D SLAM approach fusing ORB triangulation estimates and depth measurements," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–15, 2022.

[11] G. Zhou, J. Yuan, H. Gao, Q. Sun, X. Zhang, and S. Yu, "The 3D LiDAR point cloud descriptor based on structural unit soft-encoding in structured environment," *Robot*, vol. 42, pp. 641–650, Nov. 2020.

[12] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: A statistical framework," *Int. J. Mach. Learn. Cybern.*, vol. 1, nos. 1–4, pp. 43–52, Dec. 2010.

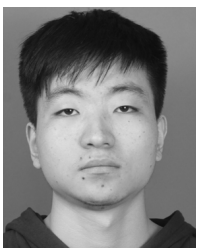
[13] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3304–3311.

[14] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *Int. J. Robot. Res.*, vol. 27, no. 6, pp. 647–665, Jun. 2008.

[15] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2006, pp. 404–417.

[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[17] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.
[18] F. Tombari, S. Salti, and L. Di Stefano, "A combined texture-shape descriptor for enhanced 3D feature matching," in *Proc. 18th IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 809–812.
[19] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 3384–3391.
[20] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.
[21] K.-Y. Huang, C.-H. Wu, and M.-H. Su, "Attention-based convolutional neural network and long short-term memory for short-term detection of mood disorders based on elicited speech responses," *Pattern Recognit.*, vol. 88, pp. 668–678, Apr. 2019.
[22] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D recognition and pose using the viewpoint feature histogram," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 2155–2162.
[23] L. He, X. Wang, and H. Zhang, "M2DP: A novel 3D point cloud descriptor and its application in loop closure detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 231–237.
[24] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 4802–4809.
[25] L. Schaupp, M. Bürki, R. Dubé, R. Siegwart, and C. Cadena, "OREOS: Oriented recognition of 3D point clouds in outdoor scenarios," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 3255–3261.
[26] K. Vidanapathirana, P. Moghadam, B. Harwood, M. Zhao, S. Sridharan, and C. Fookes, "Locus: LiDAR-based place recognition using spatiotemporal higher-order pooling," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 5075–5081.
[27] J. Yuan et al., "A novel approach to image-sequence-based mobile robot place recognition," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 9, pp. 5377–5391, Sep. 2021.
[28] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard, "Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 1249–1255.
[29] M. A. Uy and G. H. Lee, "PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4470–4479.
[30] Y. Gong, F. Sun, J. Yuan, W. Zhu, and Q. Sun, "A two-level framework for place recognition with 3D LiDAR based on spatial relation graph," *Pattern Recognit.*, vol. 120, Dec. 2021, Art. no. 108171.
[31] D. Filliat, "Interactive learning of visual topological navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 248–254.
[32] H. Gao, X. Zhang, J. Yuan, J. Song, and Y. Fang, "A novel global localization approach based on structural unit encoding and multiple hypothesis tracking," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 11, pp. 4427–4442, Nov. 2019.
[33] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Dec. 2011, pp. 2987–2992.
[34] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.



Yansong Gong received the B.Sc. and M.Sc. degrees in software engineering from Nankai University, Tianjin, China, in 2019 and 2022, respectively.

His current research interests include localization of autonomous vehicles and place recognition.

Jing Yuan received the B.Sc. degree in automatic control and the Ph.D. degree in control theory and control engineering from Nankai University, Tianjin, China, in 2002 and 2007, respectively.

From 2007 to 2018, he was with the College of Computer and Control Engineering, Nankai University, where he is currently a Professor with the College of Artificial Intelligence. His current research interests include motion planning, simultaneous localization and mapping (SLAM), and target tracking.

Fengchi Sun received the B.Sc. degree in industrial automation and the M.Sc. degree in automation from the Shandong University of Science and Technology, Qingdao, China, in 1994 and 1998, respectively, and the Ph.D. degree in control theory and control engineering from Nankai University, Tianjin, China, in 2003.

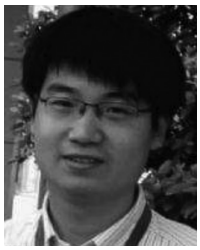
He is currently a Professor with the College of Software, Nankai University. His current research interests include autonomous mobile robots and embedded systems.

Qinxuan Sun received the B.Sc. degree in electronic information engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2013, and the M.Sc. and Ph.D. degrees in control science and engineering from Nankai University, Tianjin, China, in 2016 and 2021, respectively.

She is currently with the Beijing Research Institute, Zhejiang Lab, Beijing. Her current research interests include unmanned aerial vehicle (UAV) navigation and simultaneous localization and mapping.

Wenbin Zhu received the B.Sc. degree in software engineering from Nankai University, Tianjin, China, in 2019, where he is currently pursuing the Ph.D. degree in control science and engineering.

His current research interests include place recognition and target tracking.



Xuebo Zhang (Senior Member, IEEE) received the B.Eng. degree in automations from Tianjin University, Tianjin, China, in 2006, and the Ph.D. degree in control theory and control engineering from Nankai University, Tianjin, in 2011.

He has been with the Institute of Robotics and Automatic Information System and the Tianjin Key Laboratory of Intelligent Robotics, Nankai University, where he is currently an Associate Professor. His current research interests include mobile robotics and motion planning.