Place Classification With a Graph Regularized Deep Neural Network

Yiyi Liao, *Student Member, IEEE*, Sarath Kodagoda, *Member, IEEE*, Yue Wang, *Student Member, IEEE*, Lei Shi, *Member, IEEE*, and Yong Liu, *Member, IEEE*

***Abstract*—Place classification is a fundamental ability that a robot should possess to carry out effective human-robot interactions. In recent years, there is a high exploitation of artificial intelligence algorithms in robotics applications. Inspired by the recent successes of deep learning methods, we propose an end-to-end learning approach for the place classification problem. With deep architectures, this methodology automatically discovers features and contributes in general to higher classification accuracies. The pipeline of our approach is composed of three parts. First, we construct multiple layers of laser range data to represent the environment information in different levels of granularity. Second, each layer of data are fed into a deep neural network for classification, where a graph regularization is imposed to the deep architecture for keeping local consistency between adjacent samples. Finally, the predicted labels obtained from all layers are fused based on confidence trees to maximize the overall confidence. Experimental results validate the effectiveness of our end-to-end place classification framework in which both the multilayer structure and the graph regularization promote the classification performance. Furthermore, results show that the features automatically learned from the raw input range data can achieve competitive results to the features constructed based on statistical and geometrical information.**

***Index Terms*—Deep learning, graph regularization, place classification.**

I. Introduction

PLACE classification is an important problem in human-robot interactions and mobile robotics, which aims to distinguish differences of environmental locations and assign a label (corridor, office, kitchen, etc.) to each location [1], [2]. It allows robots to achieve spatial awareness through semantic understanding rather than having to rely on precise coordinates in communicating with humans. Furthermore, the semantic labels has the potential to efficiently facilitate other robotic functions such as mapping [3], behavior-based navigation [4], task planning [5], [6], and active object search and rescue [7]. Therefore, the task of place classification has been intensively explored in the robotics community [8]–[10].

In general, place classification is carried out through environment sensing. Laser range finders, cameras and RGB-D sensors are the mostly used sensing modalities. Location and topological information can also be informative in place classification. In this paper, we attempt to exploit both the sensory data and location information. We assume all the maps in this paper contain these two parts of information and some of the maps are labeled with human knowledge. Then the place classification problem can be stated as predicting the labels of new environments given the labeled maps.

By analyzing those two forms of data, sensory data and location information, we can gain insights into the characteristics of the place classification problem. Raw sensory data encode the environment information at different locations which can provide discriminative information between different classes. However, this requires an effective feature extraction method and most of the previous works tend to extract hand-engineered features from the raw data [11], [12]. Our opinion is that the handcrafted features may not fully exploit the potential to achieve higher generalization ability. On the other hand, locations encode spatial information of the environment and indicate local consistency of the labels, which means the positions at spatial proximity have higher probability of having the same class labels.

It is to be noted that another difficulty in place classification is the influence of different field of views (FOVs) of the sensors used. For example, the data collected by a 180° FOV laser range finder facing approximately a corner of a corridor may not contain sufficient information for classification. If the laser range finder collects 360° FOV data at a door of an office room, the robot might be confused by mixed information from two classes.

In order to address these problems, in this paper, we propose a graph regularized deep learning approach with classification on multilayer inputs. The pipeline of our system is illustrated in Fig. 1, which can be split into three parts.

1) *Construction of Multilayer Inputs:* The environmental information is represented through the generalized Voronoi graph (GVG) [13], a topological graph in which the nodes correspond to the sensory data and the edges denote the relationships in this paper. By fusing the information and eliminating the end-nodes, we

Manuscript received January 31, 2016; revised April 28, 2016; accepted June 12, 2016. Date of publication June 29, 2016; date of current version December 7, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant U1509210, in part by the Natural Science Foundation of Zhejiang Province under Grant LR13F030003, and in part by the Joint Centre for Robotics Research between Zhejiang University and the University of Technology, Sydney. *(Corresponding author: Yong Liu.)*

Y. Liao, Y. Wang, and Y. Liu are with the State Key Laboratory of Industrial Control Technology and Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: yongliu@iipc.zju.edu.cn).

S. Kodagoda and L. Shi are with the Centre for Autonomous Systems, University of Technology Sydney, Ultimo, NSW 2007, Australia.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCDS.2016.2586183

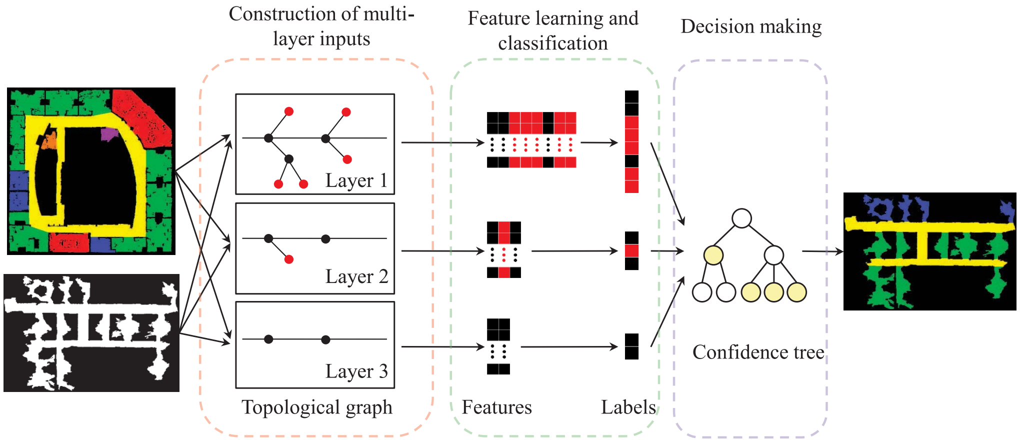


Fig. 1. Pipeline of the semisupervised learning system. Given a training map and a testing map, multilayer inputs are first constructed for both maps to represent the environmental information with different FOVs. At each layer, end-nodes (denoted as red) in the lower layer are eliminated and the information carried by them are fused to their parent node (denoted as black). Then, the raw information carried by each node is represented as a feature vector using the deep architecture, and end-to-end predicted labels are obtained. Finally, a confidence tree is constructed to fuse the predicted labels of different layers.

implement a recursive algorithm to construct multilayer inputs with hierarchical GVGs. The inputs of higher layers contain information of larger FOV, represented by increasingly succinct GVG. The features are extracted from each layer of input and classified independently.

2) *Graph Regularized Deep Architecture for Feature Learning and Classification:* We adopt the deep architecture that learns features from the raw data automatically. A graph regularizer is imposed to the deep architecture to keep the local consistency, where an adjacency graph is constructed to depict the adjacency and similarity between the samples. Our training map and testing maps are fed into the deep architecture for feature learning at the same time, which forms a semisupervised learning framework. The output of this step is the predicted labels of different layers.
3) *Confidence Tree for Decision Making:* After receiving the classification results of multilayer inputs, confidence trees are constructed according to the topological graph, and a decision making process is carried out to maximize the overall confidence.

The remainder of this paper is organized as follows. Section II reviews the related literature. In Section III, we introduce the construction of our multilayer inputs and the confidence tree for decision making. The semisupervised classification with graph regularization is given in Section IV. Experimental results are presented in Section V to validate the effectiveness of our end-to-end classification framework. Then this paper is concluded in Section VI.

II. Related Work

There are various sensors that help robots to perceive the environment, such as cameras and laser range finders. Previous works have demonstrated the effectiveness of both camera data and laser range finder data for classifying places. For example, Shi and Samarabandu [14] and Liao *et al.* [10] extracted features from the vision data, while Mozos *et al.* [11] and Sousa *et al.* [12] classified the places based on laser range data. In this paper, we focus on the place classification based on laser range data, however, our approach can be easily extended to other modality of sensors such as vision data.

Laser range finders provide nonnegative beam sequences describing range and bearing to existing obstacles within a specific range. Mozos *et al.* [11] extracted features from the 360° laser range data and those features were fed into an Adaboost classifier to label the environment. Sousa *et al.* [12] reported superior results on a binary classification task using a subset of above mentioned features, and the support vector machine (SVM) as the classifier. In our past work, we implemented a logistic regression-based classifier, as a binary and multiclass problem contributing to higher accuracies [15], [16]. The work was further extended to address the generalizability of the solution through a semisupervised place classification over a generalized Voronoi graph (SPCoGVG) [8]. Recently, Premebida *et al.* [17] proposed to combine multiple classifiers using a mixture of probabilistic models, on which a dynamic Bayesian network was constructed to incorporate the past inferences. In all of these methods, the features were extracted from the laser range data based on statistical and geometrical information, or so-called hand-engineered features. For instance, the average and the standard deviation of the beam length, the area and perimeter of the polygon specified by the observed range data and bearing were included in the feature set.

In the past decade, the unsupervised feature learning has drawn much attention as the developing of deep learning methods [18]–[20]. The deep learning methods achieved remarkable results in many areas, including object recognition [21], [22], natural language processing [23], [24], speech recognition [25], and even emotion recognition [26], which demonstrated that discovering and extracting features automatically can usually achieve better results on representation learning [27]–[29]. In the robotics community, deep learning methods also shown their outstanding performances in a range of applications [30]–[32]. Inspired by the success of unsupervised feature learning, in this article we present an end-to-end framework to solve the place classification task, where the deep learning method is employed to learn features automatically from the laser range data. A recent approach based on raw laser range data is proposed by Kaleci *et al.* [33], where the laser range data are normalized and classified based on clustering method, which can be regarded as learning features with a shallow neural network.

We also exploit the local consistency of classes with the assumption that samples located in the same small region are more likely to have the same labels. Previous research has included this particular characteristic for performance promotion and many studies were carried out with consideration of the local consistency [3], [11], [34]–[36].

In this paper, we consider the local consistency during the feature learning process, where, the features learn to keep the local invariance with a graph regularization. There is similar work on implementing the graph regularized deep learning models [37], [38]. Both [37] and [38] utilized a margin-based loss function proposed by Hadsell *et al.* [37]. These works have demonstrated the effectiveness of the graph embedding in dimensionality reduction and image classification.

Algorithm 1: Generate Higher Layer of Input From the Previous Layer

```
Input: $G^{(l)} = \{V^{(l)}, E^{(l)}\}$, the range data $r_i^{(l)}$ on each
       node $v_i^{(l)}$
Output: $G^{(l+1)} = \{V^{(l+1)}, E^{(l+1)}\}$, the range data $r_i^{(l+1)}$
        on each node $v_i^{(l+1)}$
for $v_i^{(l)} \in V^{(l)}$ do
    if $numel(N(v_i^{(l)})) > 1$ then
        Preserve $v_i^{(l)}$, i.e. $v_i^{(l+1)} = v_i^{(l)}$;
        Construct $r_i^{(l+1)}$ and $\hat{r}_i^{(l+1)}$ from $r_i^{(l)}$ and all of the
        $r_j^{(l)}$ carried by $v_j^{(l)} \in N(v_i^{(l)})$;
    end
    for $v_j^{(l)} \in N(v_i^{(l)})$ do
        if $v_j^{(l)} \in M(v_i^{(l)})$ then
            Eliminate $e_{ij}^{(l)}$ and $v_j^{(l)}$;
        else
            Preserve $e_{ij}^{(l)}$, i.e. $e_{ij}^{(l+1)} = e_{ij}^{(l)}$;
        end
    end
end
```

III. Multilayer Construction and Decision Making

In this paper, we assume a laser range finder with a typical FOV of 180°. This is a limited FOV which can give rise to many classification inaccuracies due to the lack of crucial information. However, the full FOV may also lead to misclassifications at the boundaries of the two different classes of places. Therefore, considering these problems, we propose to construct multilayer inputs for classification followed by fusion of the results.

A. Construction of Multilayer Inputs

1) Data Representation on GVG: In this paper, our multilayer inputs is represented by the hierarchical GVG [13], a topological graph which has been successfully applied to navigation, localization and mapping. The general representation of GVG is composed of meet-points (locations of three-way or more equidistance to obstacles) and edges (feasible paths between meet-points which are two-way equidistance to obstacles) [39]. GVG can be constructed with different solutions with respect to the minimum distance between obstacles. We adopt the finest available resolution (the same as in our previous work [8]) to construct the first layer GVG, where the minimum distance between obstacles is 0.075 m. Then the higher layers of GVGs are constructed to describe the environment at different levels of granularity.

Let us denote hierarchical GVGs as $\langle G^{(1)}, G^{(2)}, \dots, G^{(L)} \rangle$ with $G^{(l)} = \{V^{(l)}, E^{(l)}\}$, where L denotes the number of layer, $V^{(l)}$ denotes nodes in layer l and $E^{(l)}$ denotes edges in layer l. For each layer, the independent sensing information is carried by the nodes in $V^{(l)}$, and the local connectivity is represented by the edges in $V^{(l)}$. More specifically, each node $v_i^{(l)} \in V^{(l)}$ corresponds to a sequence of range data $r_i^{(l)}$, assigned the label $y_i^{(l)}$ for the training maps, while $e_{ij}^{(l)} \in E^{(l)}$ reveals the connection between two neighboring nodes $v_i^{(l)}$ and $v_j^{(l)}$.

The first layer $G^{(1)} = \{V^{(1)}, E^{(1)}\}$ describes the environment in most detailed level of granularity with the originally adopted laser range data. As the laser range finder is of 180° FOV with 1° angular resolution, each node $v_i^{(1)} \in V^{(1)}$ corresponds to a sequence of range data $r_i^{(1)}$ with 180 dimension.

2) Recursive Higher Layer Construction Algorithm: The construction of a higher layer GVG is implemented by fusing the information carried by connected nodes and then eliminating those marginal nodes. Algorithm 1 demonstrates the process of building higher layer GVG from a given lower layer. We make some definitions here for better explanation of the algorithm. $N(v_i)$ is defined as the directly connected neighbor set of v_i, then $v_j \in N(v_i)$ means there is an edge $e_{ij} \in E$ between v_i and v_j. In addition, numel(N) is defined as the number of elements contained in N. Then numel$(N(v_i)) = 1$ means v_i is an "end-node," i.e., the node without children. Further define $M(v_i)$ as the set of end-nodes connected to v_i, which is obviously $M(v_i) \subseteq N(v_i)$. As seen from Algorithm 1, the construction process fuses the information carried by v_i's neighbors if v_i is not an end-node (detailed fusion process is given in Section III-A3), otherwise v_i is eliminated from the higher layer.

The L layer of data can be generated by recursively applying Algorithm 1 for $L - 1$ times, which means by taking the output of the lth layer as the input of the $(l + 1)$th layer. This process can be illustrated in Fig. 2 with $L = 3$. In this example, the end-nodes are denoted as red. It is to be noted that when moving to higher layers, the

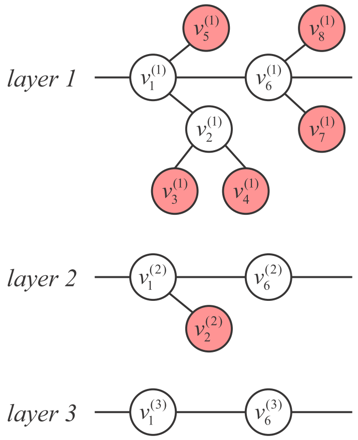


Fig. 2. Example of multilayer GVG. The end-nodes are denoted as red. The red nodes $v_1^{(1)}$, $v_2^{(1)}$, and $v_6^{(1)}$ in layer 1 are fused with their neighbors, respectively, where, $v_1^{(2)}$ is composed of $(v_1^{(1)}, v_2^{(1)}, v_5^{(1)})$, $v_2^{(2)}$ is composed of $(v_1^{(1)}, v_2^{(1)}, v_3^{(1)}, v_4^{(1)})$ and $v_6^{(2)}$ is composed of $(v_1^{(1)}, v_6^{(1)}, v_7^{(1)}, v_8^{(1)})$. Then all the red nodes are eliminated from layer 1. This process will be performed recursively on layer 2 to generate layer 3.

number of nodes in each layer decreases with the elimination of the end-nodes. More details are given in the caption of Fig. 2.

An illustration of the different $G^{(l)} = \{V^{(l)}, E^{(l)}\},\ l = 1, 2, 3$ layers constructed from a specific map is given in Fig. 3. In the first layer, the nodes are distributed more densely in the map. When approaching higher layers, the tree structure represents more and more abstract information. It is to be noted that the number of the end-nodes (denoted as red asterisks) decreases as the progression of the layers which is a consideration for choosing the $L = 3$ in our experiments.

3) Data Generation: This section describes the details about the construction of the higher-layer range data $r_i^{(l+1)}$ and $\hat{r}_i^{(l+1)}$, where the latter is generated from the former with fixed length. As stated in Algorithm 1, given $v_i^{(l)}$ satisfying numel$(N(v_i^{(l)})) > 1$ (i.e., $v_i^{(l)}$ is not end-node), range data received at the respective nodes are integrated to achieve a better perception.

Given each $v_i^{(l)}$ with numel$(N(v_i^{(l)})) > 1$, first a local map is generated using occupancy grid mapping [40] based on the respective range data in lth layer, including $r_j^{(l)}$ carried by $v_j^{(l)} \in N(v_i^{(l)})$ and $r_i^{(l)}$. This is achieved by transforming all $r_j^{(l)}$ to $r_i^{(l)}$'s coordinate frame, which assumes the knowledge of the global robot poses at all times. In this local map, a virtual scan $r_i^{(l+1)}$ is then generated by applying ray casting at position $v_i^{(l)}$ with 1° angular resolution, which is the same as the setting of the real laser range finder.

As the dimensions of the fused range data $r_i^{(l+1)}$ could be different in various nodes, linear interpolation on the data is then performed to keep same dimension of data throughout the process. This leads to an sequence $\hat{r}_i^{(l+1)}$ with fixed dimension of 360.

Acknowledging the fact that the interpolated points may not contain high information, a completeness rate, which is

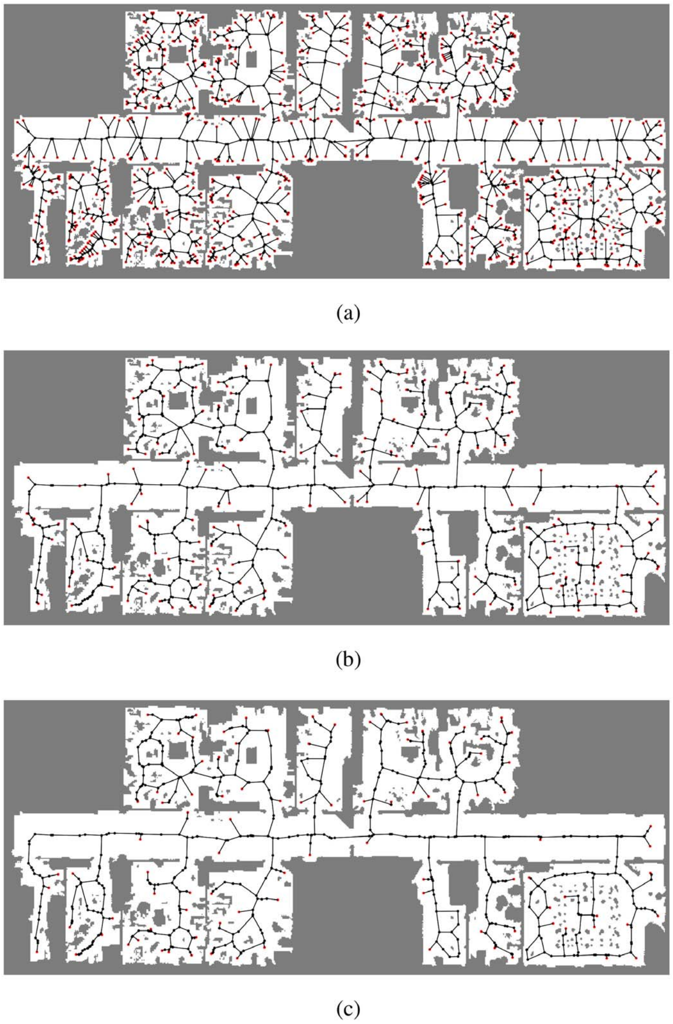


Fig. 3. Multilayer of the GVG graph $G^{(l)} = \{V^{(l)}, E^{(l)}\}, l = 1, 2, 3$ on Fr79. The red nodes correspond to the end-nodes, which will be eliminated in the next layer, and the black nodes will be preserved. The edges reveals the connection between these nodes. (a) Layer 1. (b) Layer 2. (c) Layer 3.

the proportion of the laser measured data (dimension of $r_i^{(l)}$) to the whole 360° data (dimension of $\hat{r}_i^{(l)}$) is defined as

$$q_i^{(l)} = \frac{\text{length}\left(r_i^{(l)}\right)}{\text{length}\left(\hat{r}_i^{(l)}\right)} \tag{1}$$

where $l = 2 \cdots L$. This measure is used in the decision making process which is discussed in the next section, thus we denote $q_i^{(1)} = 180/360 = 0.5$ for uniformity when $l = 1$. However, the linear interpolation is not applied to layer 1 since the initial laser range data $r_i^{(1)}$ always has the same dimension of 180, so that linear interpolation becomes unnecessary. By applying this data preprocessing approach, the laser range data in layer 2 to layer L are kept in the fixed length of 360. Note that it is always $r_i^{(l)}$ which is employed to construct the next layer, rather than the preprocessed $\hat{r}_i^{(l)}$.

As an example, Fig. 4 illustrates the construction of a sequence of input in layer 2 using the corresponding inputs in layer 1, followed by the result after linear interpolation. The details are given in the caption of Fig. 4.

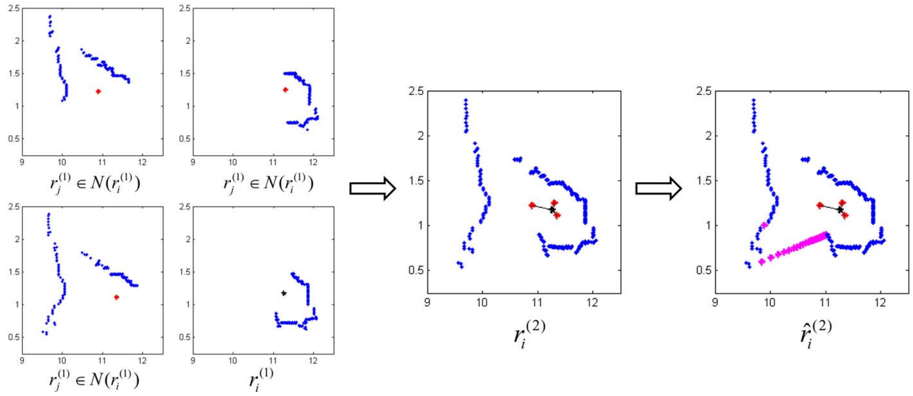


Fig. 4. Example of constructing $r_i^{(2)}$ and $\hat{r}_i^{(2)}$ where the axes are in meters. The left four figures illustrate $r_i^{(l)}$ and all of the $r_j^{(l)}$ carried by $v_j^{(l)} \in N(v_i^{(l)})$, where the black asterisk node denotes the position of $v_i^{(l)}$, the red asterisk nodes denote the position of $v_j^{(l)}$ and the blue nodes denote the range data collected from the real environment. Then the middle figure shows the constructed $r_i^{(2)}$ using ray casting. The interpolated sequence is given on the right, where the magenta points correspond to the interpolated ones. In this example, we have $q_i^{(2)} = 332/360 = 0.9222$.

B. Decision on Multilayer Results

1) Construction of the Confidence Tree: With the L layer of inputs, we can obtain the predicted labels from L independent classifiers, which can be formed to be confidence trees with L layers shown in Fig. 5(a), where each node denotes the predicted label $\hat{y}_i^{(l)}$ of $v_i^{(l)}$ and its corresponding confidence $c_i^{(l)}$. By maximizing the overall confidence of each tree structure, it is intended to obtain higher accuracy in classification.

All of these tree structures are built from the dependencies in Algorithm 1 except for some minor difference—during the construction of these tree structures, a parent node $v_i^{(l+1)}$ owns its children $v_i^{(l)}$ and $v_j^{(l)} \in M(v_i^{(l)})$, while the range data of $v_i^{(l+1)}$ is constructed from the range data carried by $v_i^{(l)}$ and $v_j^{(l)} \in N(v_i^{(l)})$. The reason is that for those nodes $v_j^{(l)} \in N(v_i^{(l)})$ and $v_j^{(l)} \notin M(v_i^{(l)})$, they are also reserved in the higher layer as $v_j^{(l+1)}$ and have their own predicted labels, so we do not consider the influence of $v_i^{(l+1)}$ to them. It is to be noted that the number of such tree structures is equal to the number of nodes left in the layer L, where the $v_i^{(L)}$ are the root nodes of these trees.

In our framework, two factors are considered when computing the confidence $c_i^{(l)}$, one is the probability $p_i^{(l)}$ obtained from the classifier for labeling $\hat{y}_i^{(l)}$ and the other is the completeness ratio $q_i^{(l)}$ obtained from the input sequence $r_i^{(l)}$ which is given in (1). Then the confidence $c_i^{(l)}$ is constructed as

$$c_i^{(l)} = p_i^{(l)} \times q_i^{(l)}. \tag{2}$$

2) Decision Algorithm: With the confidence trees denoting the predicted label $\hat{y}_i^{(l)}$ and its corresponding confidence $c_i^{(l)}$ for each given $v_i^{(l)}$, the aim of decision making is then to search the optimized confidence $c_{i*}^{(l)}$ and assign the optimized label $\hat{y}_{i*}^{(l)}$ to each node, leading to the maximum value of the overall confidence.

In each tree structure, we make decisions from children to parents while comparing two consecutive layers based on the decision Algorithm 2. It is to be noted that for the comparison between layer l and layer $l-1$, the confidence of the parent $v_i^{(l)}$ is always compared to the average optimized confidence of its children $v_j^{(l-1)}$ and we assume the optimized confidences in layer 1 are known as the original confidences. As for the optimized predicted labels, Algorithm 2 tells that they are only changed to follow their ancestor when this ancestor beats its children in confidence. In other words, if no ancestor of a leaf node gain advantages in confidence, then this leaf node would keep the initial label $\hat{y}_i^{(1)}$ as its optimized label $\hat{y}_{i*}^{(1)}$. Note that although we can obtain the optimized labels for all nodes from this decision algorithm, only the labels of the leaf nodes are exported as output since the classification performance is evaluated based on these leaf nodes. An example is given in Fig. 5(b) for better clarity.

Algorithm 2: Decision Making on the Confidence Trees

```
Input: Confidence trees where each node $v_i^{(l)}$ denotes
       the predicted label $\hat{y}_i^{(l)}$ and the corresponding
       confidence $c_i^{(l)}$.
Output: Optimized labels of leaf nodes $\hat{y}_{i*}^{(1)}$.
Initialize $c_{i*}^{(1)} = c_i^{(1)}$ and $\hat{y}_{i*}^{(1)} = \hat{y}_i^{(1)}$;
for $l = 2 \cdots L$ do
    for $v_i^{(l)} \in V^{(l)}$ do
        Average the optimized confidence of $v_i^{(l)}$'s
        children $v_j^{(l-1)}$ as $\frac{1}{n_i}\sum_j c_{j*}^{(l-1)}$;
        if $\frac{1}{n_i}\sum_j c_{j*}^{(l-1)} > c_i^{(l)}$ then
            Denote $c_{i*}^{(l)} = \frac{1}{n_i}\sum_j c_{j*}^{(l-1)}$;
        else
            Denote $c_{i*}^{(l)} = c_i^{(l)}$;
            All descendants of $v_i^{(l)}$ are assigned the label
            $\hat{y}_{i*}^{(l)}$.
        end
    end
end
```

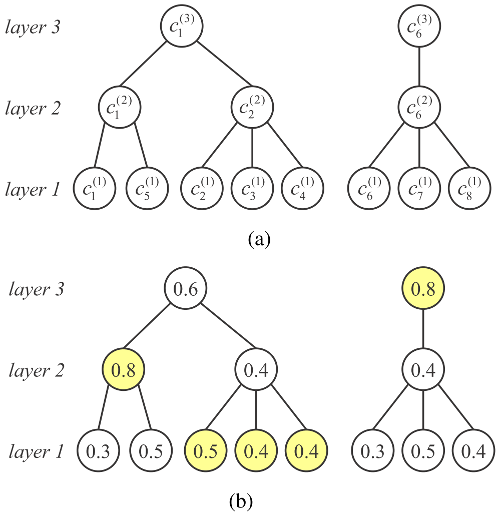


Fig. 5. Confidence trees built from Fig. 2 and a corresponding example. (a) Confidence tree: each parent node $v_i^{(l+1)}$ has children $v_i^{(l)}$ and $v_j^{(l)} \in M(v_i^{(l)})$. (b) Decision example: in this example, let us assume that the confidence of each node is known. By applying the decision method given in Algorithm 2, first we have the initialization $c_{i*}^{(1)} = c_i^{(1)}$ and $\hat{y}_{i*}^{(1)} = \hat{y}_i^{(1)}$. And then average confidence of the children in bottom most layer are compared with their corresponding parents in the immediate upper layer. In the left tree, $c_1^{(2)}$ is larger than the average value of $c_1^{(1)}$ and $c_5^{(1)}$, and therefore, $c_{1*}^{(2)} = 0.8$ and both the respective children ($v_1^{(1)}$ and $v_5^{(1)}$) are assigned the label $\hat{y}_1^{(2)}$. The $c_2^{(2)}$ is smaller than the average value of $c_2^{(1)}$, $c_3^{(1)}$, and $c_4^{(1)}$, hence, these leaf nodes remain their initial label and $c_{2*}^{(2)} = (0.5 + 0.4 + 0.4)/3 = 0.4333$. Finally, $c_1^{(3)} = 0.6$ is compared with $(c_{1*}^{(2)} + c_{2*}^{(2)})/2 = 0.6167$. Since the confidence of layer 3 is smaller than the optimized average confidence combined from layer 1 and layer 2, the final optimized confidence is $c_{1*}^{(3)} = 0.6167$ and the optimized labels do not change. By applying the same decision process on the right tree in the figure, $v_6^{(1)}$, $v_7^{(1)}$, and $v_8^{(1)}$ are labeled the same as $v_6^{(3)}$.

We can also evaluate the results obtained from those L independent classifiers separately with the help of these constructed trees. To ensure the fairness, results obtained from different layer of classifiers are all compared on the accuracy of bottom layer. Obviously, the results observed from the input of layer 1 do not need to be modified while the higher layers should spread their predicted labels to the bottom layer. Given a specific layer l ($l > 1$), all of the nodes on the bottom layer are assigned the same label as their ancestor in layer l. For example, as shown in Fig. 5(b), the $v_1^{(l)}, v_2^{(l)}, \ldots, v_5^{(l)}$ will be labeled by the $v_1^{(3)}$'s predicted label when we evaluate the results of layer 3.

IV. Semisupervised Learning and Classification

We have introduced the construction of multilayer inputs and decision making on the multilayer results in Section III. In this section, we discuss the classification problem of how to train on each layer with the input data and obtain the predicted labels of the testing maps. This is implemented by a deep learning structure, with the capability to automatically learn features from the raw input data. The L layer of inputs are trained through L independent deep learning modes as indicated in Fig. 1, though, these models have the same structure with raw laser range data being the input and predicted labels being the output as shown in Fig. 6. Thus the discussion below in this section is not confined to any specific layer and hence the superscripts are omitted. It is to be noted that our training process is semisupervised since both the training map and the testing map are employed for model training, where only the labels of the training map are available. The semisupervised learning process has the advantage of gaining richer information of data distribution, while keeping the spatial consistency as we will introduce in this chapter.

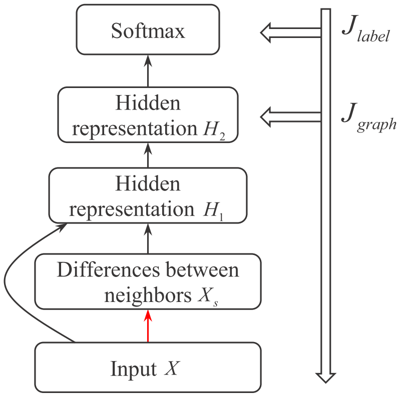

Softmax
J_{label}
Hidden representation H_2
J_{graph}
Hidden representation H_1
Differences between neighbors X_s
Input X

Fig. 6. Model training in semisupervised learning. The second layer has fixed parameters which computes the consecutive differences of our input (denoted as red). Then both the input and the output of the second layer will be fed into the latter process. For the fine-tuning process, the J_{label} is imposed to the softmax classifier and all of the parameters in the neural network (except the fixed layer) will be adjusted, while J_{graph} is imposed to the last hidden layer and will only influence the feature learning process.

A. Semisupervised Learning With Graph Regularization

In the classification problem, we denote the training pairs as ($X_l \in \mathbb{R}^{m \times n_l}$, $Y_l \in \mathbb{R}^{1 \times n_l}$) as a convention, where m denotes the input dimension, n_l denotes the number of training samples. Particularly, each column in X_l is a sequence of laser range data r, i.e., $x_l^i = r_i$. The testing data can be defined in the same way as $X_u \in \mathbb{R}^{m \times n_u}$, where n_u denotes the number of testing samples. Then the task of the classification problem is to obtain predicted labels of X_u given X_l and Y_l. In addition, we denote $X = [X_l \;\; X_u] \in \mathbb{R}^{m \times n}$ as the combination of training data and testing data with $n = n_l + n_u$, since X is fed into the model as a whole during our semisupervised training process.

As illustrated in Fig. 6, the input is first fed into a set of fixed parameters (denoted as red) to compute the differences between the consecutive beams in each raw scan, as the consecutive differences can also provide rich information to the place classification and is often employed for extracting geometric features in the previous works [11], [12]. In the practical experiments, we sort both of the input and consecutive differences to guarantee the rotational invariance.

From this point on, both the input and output of this fixed layer are fed into the stacked auto-encoders for feature learning. Auto-encoder is the widely used structure for building deep architectures, which is composed of an encoder and a decoder. By feeding the representation learned from the previous encoder as the input into another auto-encoder, we can obtain the stacked hidden representations as shown in Fig. 6. Let us denote sigmoid function as $f(x) = 1/(1 + e^{-x})$, then the ith layer of encoder and decoder can be represented as follows:

$$\begin{aligned} H_i &= f(W_i H_{i-1} + b_i) \\ \hat{H}_{i-1} &= f\left(W_i^T H_i + c_i\right) \end{aligned} \tag{3}$$

where H_{i-1} and $\hat{H}_{i-1}$ denote the input and its reconstruction, H_i denotes the hidden representation and W_i, b_i, c_i denote the weighted parameters, respectively.[1] In this paper, the weights in each pair of encoder and decoder are tied together as shown in (3).

For each layer of auto-encoder, the unsupervised pretraining is applied to obtain better parameters than random initialization [18] by minimizing the reconstruction cost

$$J_{\text{pre}} = \frac{1}{m}\left\|H_{i-1} - \hat{H}_{i-1}\right\|_F^2. \tag{4}$$

Note that the decoder is discarded after pretraining while the encoder is preserved. The hidden representation learned by the last auto-encoder can be regarded as the feature for the input to the classifier.

In this paper reported here, the softmax classifier is applied to the features learned from stacked auto-encoders for classification, which is formulated as follow:

$$p_i = \frac{\exp\left(w_i^T h\right)}{\sum_j \exp\left(w_j^T h\right)} \tag{5}$$

where p_i corresponds to the probability that the hidden representation vector h belongs to the ith class.

After pretraining and classification, back propagation can be used to fine-tune the whole learning process for further promotion, which means the parameters of preserved encoders and softmax are trained together. In order to keep the local consistency, we add a graph regularization term during fine-tuning to learned representation. The cost function of the fine-tuning is given as follow:

$$\begin{aligned} J_{\text{fine}} &= J_{\text{label}} + J_{\text{graph}} \\ &= \frac{1}{n_l}\sum_{i=1}^{n_l} J_{\text{label}}\left(x_l^i, y_l^i\right) + \frac{\lambda}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij}\left\|h_i - h_j\right\|^2 \end{aligned} \tag{6}$$

where the first term corresponds to the prediction error of the training data, and the second term is the graph regularization. Here h_i and h_j are the outputs of the last hidden layer with respect to the inputs x_i and x_j (x_i and x_j are two arbitrary columns in X), and s_{ij} is the similarity measurement between the samples x_i and x_j that are connected in GVG, which is an element of the adjacency graph $S = [s_{ij}]_{n\times n}$. Fig. 6 also illustrates the way our cost function work. The costs caused by the prediction error is imposed on the softmax classifier and then our graph regularization is imposed on the last hidden layer. Therefore during the fine-tuning the J_{label} influences all of the parameters, while J_{graph} influences parameters except for the softmax classifier.

B. Graph Regularization in Place Classification Problem

As shown in (6), the learned features h_i and h_j with large weight s_{ij} will be pushed together with the graph regularization term. In this section, we describe the details about the construction of the adjacency graph S which can be built in two steps. Firstly we define the connected relationships between samples and then calculate their weights of the connected edges.

In the place classification problem, the connected relationships in the topological graph GVG are directly employed to the adjacency graph. Then the samples with close coordinates are forced to be represented by the features with close distances. As for the weights which corresponds to the strength of the graph regularization, it is inversely associated with two distances, i.e., the distance between coordinates and the distance between the input data, which can be formulated as

$$s_{ij} = \frac{\alpha}{d_{ij}} + \frac{\beta}{\left\|x_i - x_j\right\|^2} \tag{7}$$

where α and β are constant weights, d_{ij} denotes the Euclidean distance between the sample coordinates. The second term defines the Euclidean distance between the input data. This weighting scheme dose not only evaluate the geometrical information, but also considers the closeness between inputs. For example, given an edge that connects two nodes belonging to corridor and office, respectively, although d_{ij} is small, $\|x_i - x_j\|^2$ can be large. Therefore, these two nodes are not forced to be too close in the representation space and the discriminative information could be preserved.

V. Experiments

To validate the effectiveness of our end-to-end multilayer learning system, we conduct experiments on six data sets collected from six international university indoor environments using (including the Centre for Autonomous Systems at the University of Technology, Sydney, several buildings in the University of Freiburg, the German Research Centre for Artificial Intelligence in Saarbruecken, and the Intel Laboratory in Seattle). On each real grid map, a simulated robot collected laser range data using a virtual on-board 2-D laser range finder (with a maximum range of 30 meters and a horizontal FOV of 180°) at the previous mentioned GVG nodes.

It is to be noted that the classes defined by humans can be somewhat vague and plentiful according to the different functions of places. However, the 2-D range data do not contain enough discriminative information to classify all these human-designed classes. Therefore, we consider three target classes as: class 1—space designed for a small number of individuals including cubicle, office, printer room, kitchen, bathroom, stairwell and elevator; class 2—space for group activities including meeting room and laboratory; and class 3—corridor.

Among these six data sets, two of them (Fr79 and Intellab) contain all of the three classes but the others contain only parts of these classes. We conduct leave-many-out training, which means one data set is utilized for training and others are used

[1] When i=1, H_{i-1} is the raw input—the combination of X and its consecutive differences X_s.

TABLE I
MULTILAYER RESULTS TRAINED ON INTELLAB

Map	L1(%)	L2(%)	L3(%)
UTS	85.20	89.49	91.24
SarrB	86.55	87.64	91.32
FrUA	86.23	92.96	91.69
FrUB	90.29	98.87	99.84
Fr79	81.99	85.87	87.90
Average	*86.05*	*90.97*	*92.40*

TABLE II
MULTILAYER RESULTS TRAINED ON FR79

Map	L1(%)	L2(%)	L3(%)
UTS	81.70	85.99	89.93
SarrB	84.16	95.44	90.46
FrUA	90.43	94.70	96.91
FrUB	88.67	98.87	99.51
Intellab	72.55	79.81	82.73
Average	*83.50*	*90.96*	*91.91*

TABLE III
MULTILAYER RESULTS TRAINED ON INTELLAB WITH GRAPH REGULARIZATION

Map	L1(%)	L2(%)	L3(%)
UTS	83.54	87.3	92.29
SarrB	89.59	96.31	90.89
FrUA	91.48	91.77	96.68
FrUB	89.97	99.19	99.84
Fr79	83.96	86.12	88.65
Average	*87.71*	*92.14*	*93.67*

TABLE IV
MULTILAYER RESULTS TRAINED ON FR79 WITH GRAPH REGULARIZATION

Map	L1(%)	L2(%)	L3(%)
UTS	80.47	89.23	90.02
SarrB	87.20	96.75	95.23
FrUA	91.06	96.12	97.47
FrUB	89.48	98.87	99.51
Intellab	73.00	79.89	82.51
Average	*84.24*	*92.17*	*92.95*

for testing. Therefore, we obtained two groups of results by training on Fr79 and Intellab, respectively.

The number of layers, L was chosen to be 3 based on the difference between two consecutive layers, which is measured by the distance between their mean completeness rates. The mean completeness of layer l is given as follows:

$$\overline{q}^{(l)} = \sum_i q_i^{(l)} \tag{8}$$

where $q_i^{(l)}$ is the completeness rate of node i in layer l as defined in (1). If the distance between $\overline{q}^{(l)}$ and $\overline{q}^{(l+1)}$ is infinitesimal, then their corresponding laser measured data in these two layers are very close. Therefore, it is noninformative to consider the higher layer $l + 1$. Taking the map Fr79 as an example, it has $\overline{q}^{(1)} = 0.50$, $\overline{q}^{(2)} = 0.91$, $\overline{q}^{(3)} = 0.98$ and $\overline{q}^{(4)} = 0.99$. Since $\overline{q}^{(4)}$ is very close to $\overline{q}^{(3)}$, we choose $L = 3$ in this paper. The dimension of each layer of the deep architecture in Fig. 6 was set as $m - m - 100 - 24 - 3$ given the input $X \in \mathbb{R}^{m \times n}$. The input and the consecutive layer have the same dimension as the raw input m, where $m = 180$ for $L = 1$ and $m = 360$ for $L = 2, 3$. For the following hidden layers, we set the dimension of the hidden representation H_1 as 100 based on experiments. The dimension of the hidden representation H_2 is set to be 24, which is the same as the dimension of the handcrafted features in [8] for fair comparison. Finally the output of our model represents a probabilistic measure of data belonging to each class. Thus the output dimension is the same as the number of our classes.

A. Multilayer Results Without Graph Regularization

We first conduct experiments to evaluate the performance of our multilayer inputs. Tables I and II show the leave-many-out classification results training on Intellab and Fr79, respectively. It is to be noted that the graph regularization is not considered here and therefore, i.e., $\lambda = 0$ in the cost function (6). In general, results of higher layers are better than that of lower layers due to the richer information contained in each node on the higher layers.

B. Multilayer Results With Graph Regularization

We also carried out experiments to validate the effectiveness of the graph regularization. The algorithm remains the same as previous settings, however, we changed the value of $\lambda = 1$ to add the graph regularization. In this experiments, we pay more attention to the geometrical neighborhood, thus we use $\alpha = 2/3$ and $\beta = 1/3$ in (7) for the construction of the adjacency graph. The classification results are shown in Tables III and IV, which are trained on Intellab and Fr79, respectively. The results have the similar trends as in Tables I and II, where higher layers give rise to better accuracies. Further comparisons of Tables I and III show that the feature learning with graph regularization performs better than without it. It reveals that the graph regularization has the advantage of improving classification performances by keeping the local consistency.

C. Fusion Results

On top of that, we show the accuracies of the multilayer graph regularized method with fusion in the second column of Tables V and VI. When compared with the results of each single layer as shown in Tables III and IV, the fusion results achieved better accuracies. For the results trained on Intellab, the average accuracy of fusion results risen to 94.02% from $L1$: 87.71%, $L2$: 92.14%, and $L3$: 92.66%, and the results trained on Fr79 also reached 93.59% from $L1$: 84.24%, $L2$: 92.17%, and $L3$: 92.95%. The fused test results trained on Intellab are diagrammatically illustrated in Fig. 7. It is to be noted that confusions between class 1 (office room and other rooms) and class 2 (meeting room) account for the major misclassifications especially in the test map of Fr79. The cause might be that class 1 is featured with narrow environment including massive clutters while the class 2 is featured with relatively larger spaces, therefore the corners of meeting room are mostly classified as office room and other rooms and some center positions of office room are assigned as office room.

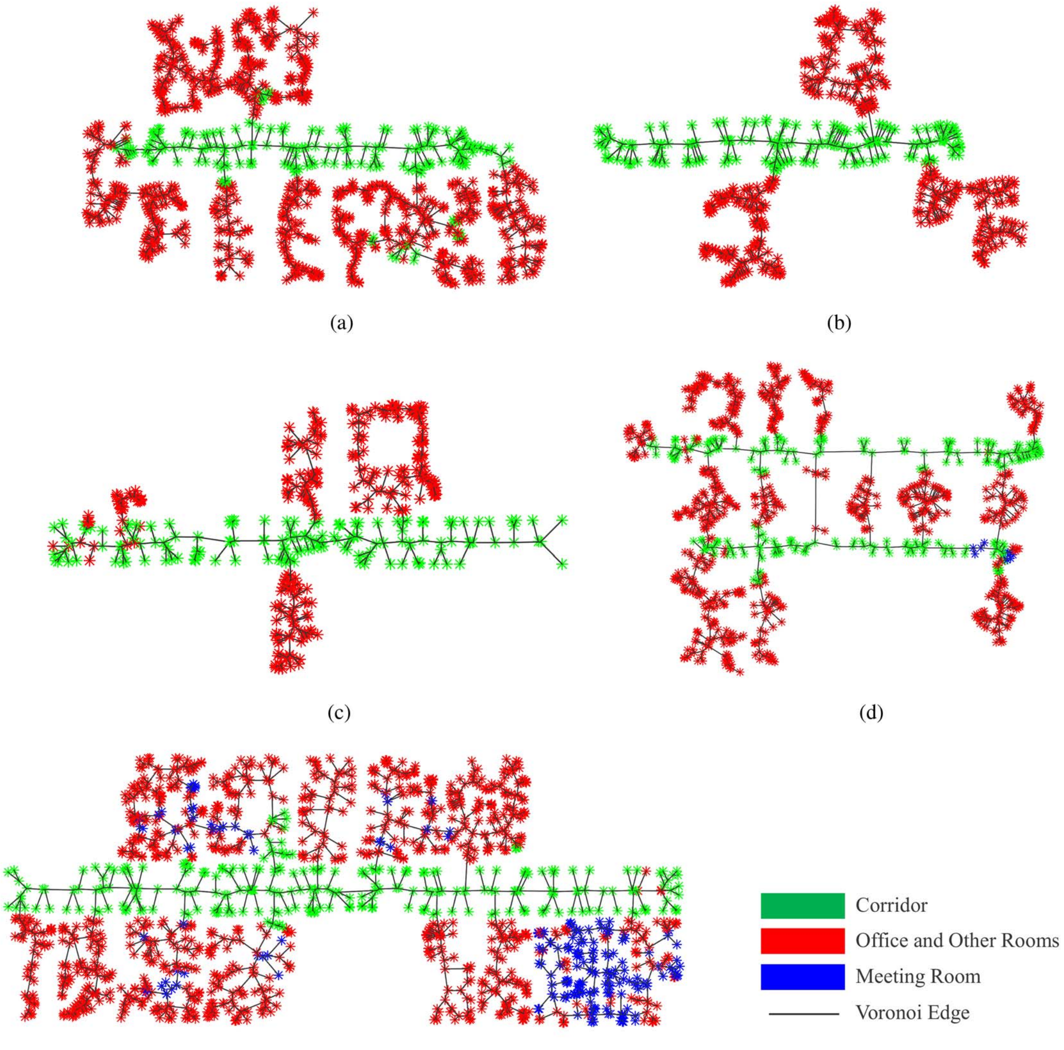
Through the experiments above, one can see that our framework is beneficial from the following two aspects.

TABLE V
GRAPH REGULARIZED FUSION RESULTS TRAINED ON INTELLAB, WITH COMPARISON TO THE STATE-OF-THE-ART

Map	Multi-layer fusion(%)	SVM(%)	SPCoGVG(%)	LVQ Mar.(%)	DBMM 180°(%)	DBMM 360°(%)
UTS	**91.24**	87.74	90.72	82.75	88.88	88.70
SarrB	**96.53**	85.68	88.72	75.49	79.83	86.77
FrUA	95.02	96.04	**96.52**	89.95	94.38	95.89
FrUB	**99.84**	97.25	98.71	83.33	95.96	97.41
Fr79	89.76	88.34	92.04	77.85	89.88	**93.28**
Average	***94.48***	*91.01*	*93.39*	*81.87*	*89.79*	*92.41*

TABLE VI
GRAPH REGULARIZED FUSION RESULTS TRAINED ON FR79, WITH COMPARISON TO THE STATE-OF-THE-ART

Map	Multi-layer fusion(%)	SVM(%)	SPCoGVG(%)	LVQ Mar.(%)	DBMM 180°(%)	DBMM 360°(%)
UTS	**90.54**	83.54	89.84	88.97	87.22	85.46
SarrB	**98.27**	82.43	93.71	92.84	91.97	86.99
FrUA	97.23	92.72	**97.71**	96.12	95.97	96.84
FrUB	**99.51**	80.74	99.19	97.89	96.93	97.09
Intellab	82.40	79.89	**86.89**	57.19	83.22	85.36
Average	***93.59***	*83.86*	*93.47*	*86.60*	*91.06*	*90.35*



(a)

(b)

(c)

(d)

(e)

Fig. 7. Test results corresponding to Table V, the GVG nodes are labeled with the graph regularized fusion results trained on Intelmap. (a) FrUA, Acc = 95.02%. (b) FrUB, Acc = 99.84%. (c) SarrB, Acc = 96.53%. (d) UTS, Acc = 91.24%. (e) Fr79, Acc = 89.76%.

1) By regularizing the deep architecture with the adjacency graph, the learned features are expected to be learned with spatial invariance intrinsically, while the hand-crafted features not able to encoding the relationship between neighboring nodes. The automatic feature learning also releases the human labor from feature designing, enabling the end-to-end learning.

2) The better discriminative performance is achieved by fusion on the confidence tree, which comprehensively considers the results from different layers. The information can be insufficient in the lower layer due to narrow FOV. In addition, in higher layers, as the FOV may cover more than one place, the multiple labels may confuse the classifier. The tree decision avoids the two problems to some extent.

D. Algorithm Comparison

To further validate the effectiveness of the proposed algorithm, it is compared to one baseline method as well as three state-of-the-art methods, including the followings.

1) *SVM:* It is employed as a baseline classifier. It is based on 24-D hand-engineered features as implemented in [8]. Specifically, 21 features are constructed based on statistical and geometrical information from the 180° FOV raw range data, which is a subset of the features used in [11]. The other three features are constructed to describe the time domain information follow [41].
2) *SPCoGVG:* SPCoGVG [8], which is composed of SVM and conditional random field to ensure the generalization ability. It also uses the 24-D hand-engineered feature the same as the SVM.
3) *LVQ Mar:* This method uses learning vector quantization (LVQ) for classification and Markov Model to incorporate the past inference [33]. Similar to this paper, they take raw laser scans as input, while their input can only be the calibrated 360° FOV range data. LVQ can be regarded as a shallow neural network with one hidden layer, thus the dimension of the hidden layer is set to be 24, which is the same as the dimension as SVM, SPCoGVG, and our method. To build the time sequence, we generate a moving trajectory based on our GVG nodes, on which the Markov Model is applied. We reimplement their method for comparison.
4) *DBMM:* Dynamic bayesian mixture models (DBMM) [17] is a method that combines two SVM classifiers using a mixture of probabilistic models, and incorporates past inferences using a dynamic Bayesian process. It is also based on hand-engineered features [11], while a larger subset with 50 dimensions is considered in this paper. The features are extracted from both 180° FOV and 360° FOV laser scan data for comparison. A time sequence is also required to build the dynamic Bayesian network, which is constructed the same as the moving trajectory in the LVQ Mar. We implement their method for comparison.

As can be seen from Tables V and VI, our method achieves superior results compared to the others. Specifically,

1) Our method and LVQ Mar. both employ the raw data to learn the features. However, LVQ Mar. is not comparable to SVM while ours outperforms SVM considerably. Particularly, LVQ performs poor when trained on FR79 and tested on Intellab as shown in Table VI. The main reason is that the structure of class 1 (corridors) at Intellab is slightly different than that of the other maps, and LVQ is not able to capture the variance with its features learned from the shallow architecture. As a result, the average performance of LVQ is even poorer when trained on Intellab and tested on the others maps in Table V. The result indicates that our method learns more generalizable features by owning the deep architecture and considering the spatial consistency. With the shallow architecture, LVQ Mar. cannot generate a good performance even it models the temporal information (spatial consistency in moving direction).
2) DBMM is expected to outperform SVM as it fuses two SVMs with two groups of hand-crafted features. Though it outperforms SVM in Table VI, the DBMM with 180° FOV achieves low accuracy on SarrB in Table V. In this case, the relatively poor accuracy is generated since one group of hand-crafted features in DBMM leads to severe misclassification between the class 1 (office and other rooms) and class 3 (corridor) with accuracy of 77.87%, while the other group of features achieves accuracy at 85.03%. Note that these two classes are both considered in their paper [17]. This result validates that the hand-crafted features should be carefully chosen to serve different maps. It is also worth to mention that DMBB with 180° FOV outperforms the DBMM with 360° FOV in Table VI, especially on the test SarrB, while in Table V the DBMM with 360° FOV generally performs better. This fact reflects that only considering the sensor information in one resolution can be insufficient to make a comprehensive classification.
3) Another possible reason that our method and SPCoGVG outperform LVQ Mar. and DBMM is that, the completed spatial consistency modeling by the adjacency graph can better constrain the classifier than the temporal consistency modeling by moving trajectory. For example, when the robot moves from classes 1 to 2, LVQ Mar. and DBMM only consider the past inferences of the nodes with class 1, while our method and SPCoGVG additionally take the inference of consequent nodes with class 2 into consideration. Besides, our method is slightly better than SPCoGVG, indicating that considering the constraints, e.g., spatial consistency, in feature learning step may be better than in classification step, as the former occurred in nonlinear feature space, while the latter utilizes the simple weighted sum.

In summary, the proposed method achieves state-of-the-art performance by using the spatial proximity aided feature learning and confidence tree based multilayer fusion, without referring to the hand-crafted features. Besides, we think a more valuable insight is that the hierarchical feature learning and decision with incorporating of prior knowledge can be leveraged in other robotics applications, as there are many structured or hierarchical priors in robot cognitive and decision tasks.

VI. Conclusion

In this paper, we presented an end-to-end place classification framework. We implemented a multilayer learning framework, including the construction of multilayer inputs and decision making on the multilayer results. Each layer of inputs were fed into a semisupervised model for feature learning and classification, which guaranteed the local consistency with a graph regularization.

Experimental results showed that the higher layer input data led to higher classification accuracy, which validated the effectiveness of the multilayer structure. By performing the semisupervised learning with or without graph regularization, we also showed that graph regularization helps promoting the classification performance by incorporating the local consistency. Furthermore, the fusion results based on the confidence tree achieved comparable results to the state-of-the-art method. In a nutshell, we achieved the generalization ability and preserved the local consistency in our end-to-end place classification framework. Future work is to apply our framework on other type of sensor data, such as RGB-D data, which have more representative and discriminative ability.

References

[1] L. Yuan, K. C. Chan, and C. G. Lee, "Robust semantic place recognition with vocabulary tree and landmark detection," in *Proc. IROS Workshop Active Semant. Percept. Object Search Real World*, 2011.

[2] A. Pronobis and P. Jensfelt, "Hierarchical multi-modal place categorization," in *Proc. ECMR*, Örebro, Sweden, 2011, pp. 159–164.

[3] A. Ranganathan and J. Lim, "Visual place categorization in maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, San Francisco, CA, USA, 2011, pp. 3982–3989.

[4] A. Poncela, C. Urdiales, B. Fernández-Espejo, and F. Sandoval, "Place characterization for navigation via behaviour merging for an autonomous mobile robot," in *Proc. 14th IEEE Mediterr. Electrotech. Conf. (MELECON)*, Ajaccio, France, 2008, pp. 350–355.

[5] C. Galindo, J.-A. Fernández-Madrigal, J. González, and A. Saffiotti, "Robot task planning using semantic maps," *Robot. Auton. Syst.*, vol. 56, no. 11, pp. 955–966, 2008.

[6] H. Çelikkanat *et al.*, "Learning context on a humanoid robot using incremental latent Dirichlet allocation," *IEEE Trans. Cogn. Develop. Syst.*, vol. 8, no. 1, pp. 42–59, Mar. 2015.

[7] A. Aydemir, K. Sjöö, J. Folkesson, A. Pronobis, and P. Jensfelt, "Search in the real world: Active visual object search based on spatial relations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, 2011, pp. 2818–2824.

[8] L. Shi and S. Kodagoda, "Towards generalization of semi-supervised place classification over generalized Voronoi graph," *Robot. Auton. Syst.*, vol. 61, no. 8, pp. 785–796, 2013.

[9] N. Sünderhauf *et al.*, "Place categorization and semantic mapping on a mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA),* Stockholm, Sweden, 2016, pp. 5729–5736.

[10] Y. Liao, S. Kodagoda, Y. Wang, L. Shi, and Y. Liu, "Understand scene categories by objects: A semantic regularized scene classifier using convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA),* Stockholm, Sweden, 2016, pp. 2318–2325.

[11] O. M. Mozos, C. Stachniss, and W. Burgard, "Supervised learning of places from range data using adaboost," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Barcelona, Spain, 2005, pp. 1730–1735.

[12] P. Sousa, R. Araújo, and U. Nunes, "Real-time labeling of places using support vector machines," in *Proc. IEEE Int. Symp. Ind. Electron. (ISIE)*, Vigo, Spain, 2007, pp. 2022–2027.

[13] H. Choset and J. Burdick, "Sensor based planning. I. The generalized Voronoi graph," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2. Nagoya, Japan, 1995, pp. 1649–1655.

[14] W. Shi and J. Samarabandu, "Investigating the performance of corridor and door detection algorithms in different environments," in *Proc. Int. Conf. Inf. Autom. (ICIA)*, Shandong, China, 2006, pp. 206–211.

[15] L. Shi, S. Kodagoda, and G. Dissanayake, "Laser range data based semantic labeling of places," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Taipei, Taiwan, 2010, pp. 5941–5946.

[16] L. Shi, S. Kodagoda, and G. Dissanayake, "Multi-class classification for semantic labeling of places," in *Proc. 11th Int. Conf. Control Autom. Robot. Vis. (ICARCV)*, 2010, pp. 2307–2312.

[17] C. Premebida, D. R. Faria, F. A. Souza, and U. Nunes, "Applying probabilistic mixture models to semantic place classification in mobile robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Hamburg, Germany, 2015, pp. 4265–4270.

[18] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[19] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.

[22] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Santiago, Chile, 2015, pp. 1440–1448.

[23] R. Collobert *et al.*, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, no. 8, pp. 2493–2537, 2011.

[24] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2014, pp. 3104–3112.

[25] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, no. 1, pp. 30–42, Jan. 2012.

[26] W.-L. Zheng and B.-L. Lu, "Investigating critical frequency bands and channels for eeg-based emotion recognition with deep neural networks," *IEEE Trans. Auton. Mental Develop.*, vol. 7, no. 3, pp. 162–175, Sep. 2015.

[27] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proc. ICML*, Bellevue, WA, USA, 2011, pp. 833–840.

[28] S. Rifai *et al.*, "Higher order contractive auto-encoder," in *Proc. Eur. Conf. Mach. Learn. Knowl. Disc. Databases—Volume Part II (ECML PKDD)*, Athens, Greece, 2011, pp. 645–660.

[29] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Comput.*, vol. 23, no. 7, pp. 1661–1674, Jul. 2011.

[30] R. Hadsell *et al.*, "Deep belief net learning in a long-range vision system for autonomous off-road driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nice, France, 2008, pp. 628–633.

[31] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *Int. J. Robot. Res.*, vol. 34, nos. 4–5, pp. 705–724, 2015.

[32] O. Sigaud and A. Droniou, "Towards deep developmental learning," *IEEE Trans. Cogn. Develop. Syst.*, vol. 8, no. 2, pp. 99–114, Jun. 2016.

[33] B. Kaleci, C. M. Şenler, H. Dutağaci, and O. Parlaktuna, "A probabilistic approach for semantic classification using laser range data in indoor environments," in *Proc. Int. Conf. Adv. Robot. (ICAR)*, Istanbul, Turkey, 2015, pp. 375–381.

[34] A. Pronobis and P. Jensfelt, "Large-scale semantic mapping and reasoning with heterogeneous modalities," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, St. Paul, MN, USA, 2012, pp. 3515–3522.

[35] Ó. M. Mozos, R. Triebel, P. Jensfelt, A. Rottmann, and W. Burgard, "Supervised semantic labeling of places using information extracted from sensor data," *Robot. Auton. Syst.*, vol. 55, no. 5, pp. 391–402, 2007.

[36] A. Ranganathan, "PLISS: Detecting and labeling places using online change-point detection," in *Proc. Robot. Sci. Syst.*, Zaragoza, Spain, 2010, pp. 185–192.

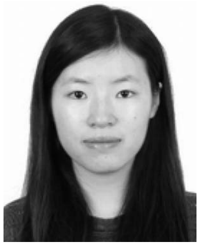
[37] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2. New York, NY, USA, 2006, pp. 1735–1742.

[38] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*. Heidelberg, Germany: Springer, 2012, pp. 639–655.

[39] T. Tao, S. Tully, G. Kantor, and H. Choset, "Incremental construction of the saturated-GVG for multi-hypothesis topological SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, 2011, pp. 3072–3077.

[40] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[41] B. Hjorth, "EEG analysis based on time domain properties," *Electroen. Clin. Neurophysiol.*, vol. 29, no. 3, pp. 306–310, 1970.



Yiyi Liao (S'15) received the B.S. degree in automation from Xi'an Jiaotong University, Xi'an, China, in 2013. She is currently pursuing the Ph.D. degree with the Institute of Cyber-Systems and Control, Department of Control Science and Engineering, Zhejiang University, Hangzhou, China.

Her current research interests include machine learning, computer vision, and mobile robotics.

Sarath Kodagoda (M'10) received the B.Sc.Eng. (Hons.) degree in electrical engineering from the University of Moratuwa, Moratuwa, Sri Lanka, in 1995, and the M.Eng. and Ph.D. degrees in robotics from Nanyang Technological University, Singapore, in 2000 and 2004, respectively.

He was a Design Engineer with a reputed multinational company. He is currently an Associate Professor and a Coordinator of the Mechanical and Mechatronics Program, University of Technology Sydney, Ultimo, NSW, Australia. His current research interests include autonomous road vehicles, human robot interaction, infrastructure robotics, machine learning, perception, target tracking, and mobile robotics.

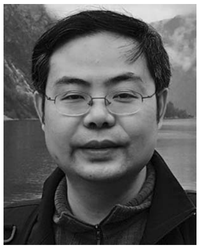
Prof. Kodagoda is an Associated Editor, a Programme Committee Member, and a regular reviewer of a number of top robotics journals and conference proceedings.

Yue Wang (S'16) received the B.Sc. degree in communication engineering from the Zhejiang University of Technology, Hangzhou, China, in 2011. He is currently pursuing the Ph.D. degree in control science and engineering with Zhejiang University, Hangzhou, under the supervision of Prof. R. Xiong.

He is a joint Ph.D. student with Stanford University, Stanford, CA, USA, funded by China Scholarship Council. His current research interests include mobile robot, machine learning, and big data analysis.

Lei Shi (M'10) received the B.Eng. degree in electrical engineering and automation from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2004, the M.Eng. degree in control systems from the University of Sheffield, Sheffield, U.K., in 2005, and the Ph.D. degree in robotics from the Centre for Autonomous Systems, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW, Australia.

He has been involved in research activities and projects related to human–robot interaction, autonomous cars, biomedical signal processing, localization and mapping, and condition assessment in the water industry. His current research interests include machine learning, applications of probability and statistics, conditional assessment, and sensors and signal processing.



Yong Liu (M'11) received the B.S. degree in computer science and engineering and the Ph.D. degree in computer science from Zhejiang University, Zhejiang, China, in 2001 and 2007, respectively.

He is currently an Associate Professor with the Institute of Cyber-Systems and Control, College of Control Science and Engineering, Zhejiang University. He has published over 30 research papers in machine learning, computer vision, information fusion, and robotics. His current research interests include machine learning, robotics vision, and information fusion.