Eyes in the Dark: Distributed Scene Understanding for Disaster Management

Eyes in the Dark: Distributed Scene Understanding for Disaster Management

Liangzhi Li, *Student Member, IEEE,* Kaoru Ota, *Member, IEEE,* Mianxiong Dong, *Member, IEEE,* Wuyunzhaola Borjigin, *Student Member, IEEE*

Abstract—Robotic is a great substitute for human to explore the dangerous areas, and will also be a great help for disaster management. Although the rise of depth sensor technologies gives a huge boost to robotic vision research, traditional approaches cannot be applied to disaster-handling robots directly due to some limitations. In this paper, we focus on the 3D robotic perception, and propose a view-invariant Convolutional Neural Network (CNN) Model for scene understanding in disaster scenarios. The proposed system is highly distributed and parallel, which is of great help to improve the efficiency of network training. In our system, two individual CNNs are used to, respectively, propose objects from input data and classify their categories. We attempt to overcome the difficulties and restrictions caused by disasters using several specially-designed multi-task loss functions. The most significant advantage in our work is that the proposed method can learn a view-invariant feature with no requirement on RGB data, which is essential for harsh, disordered and changeable environments. Additionally, an effective optimization algorithm to accelerate the learning process is also included in our work. Simulations demonstrate that our approach is robust and efficient, and outperforms the state-of-the-art in several related tasks.

Index Terms—Distributed artificial intelligence, scene understanding system, disaster management.

1 Introduction

Disaster is a persistent challenge to the human world. In recent years, lots of new technologies have emerged to improve our responses to various disasters. Although many encouraging researches have been made, one grave problem remains unsolved, i.e., how to safely explore damaged architectures or other dangerous areas. In fact, there are so many tragedies caused by secondary disasters that people cannot overlook this problem any more. Robotic, which is a great substitute for humans to carry out dangerous tasks, becomes the first choice in these scenarios.

Since robots require a sufficient understanding of the surrounding environment before any movements, scene understanding ability is of great importance for robots to be competent for these tasks. However, traditional 2D methods struggle in such situations, because they give little space information, which is very important for robots to move around and interact with the world. Therefore, 3D perception will become a vital topic in robotic for disaster management. While many approaches have been proposed in the field of 3D object detection and recognition, all these researches cannot be applied to the disaster-handling robots directly, due to some limitations existing in such situations. To our best knowledge, few researchers are working on this problem.

The obstacles lying between the disaster-handling robots and existing 3D scene understanding methods mainly include the following facts. Firstly, most of the existing approaches require the RGB information as well as the depth image. Although depth sensors are able to output stable and robust 3D images regardless of illumination conditions, it is very difficult for RGB cameras to take a picture in lower light situations [1], which, however, is very common in many disaster scenarios. And, RGB cameras can hardly obtain consistent images in different illumination conditions. Therefore, all the 3D scene understanding methods requiring RGB data are not suited for disaster-handling robots. Secondly, the Field of View (FOV) of robots is very limited when exploring the damaged house. Many objects, e.g. furniture and electronics, maybe tilted or even overturned due to some external forces. In this situation, the on-board sensors may get the objects' image from some uncommon or unanticipated view. Since it is very difficult to recognize the objects from some specific views [2], the performance of traditional methods is usually unsatisfied. Another problem is, learning-based perception systems require a great number of data in their training process, which is a huge challenge for traditional standalone approach [3].

To solve these problems, we desire to propose a Convolutional Neural Network (CNN) based robotic 3D scene understanding method to improve robots' feasibility and adaptability in disaster scenarios, as shown in Fig. 1. A robot equipped with depth sensors is sent to explore dangerous environments. With the ability of 3D scene understanding, robots can deal with various conditions and give great help to disaster preparedness, relief and recovery. Every robot is self-contained, but can also share its information and understandings in the cloud platform. The proposed system is highly distributed and parallel, which is of great help to improve the efficiency of system training. Each robot has a CNN network installed in its firmware. Robots can upload its captured data to the nearby server for further fine-tuning, and the updated parameters will be transferred back to the robots to improve their adaptability to their current

• *Authors are with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan. E-mail:{16096502, ota, mxdong, 16096505}@mmm.muroran-it.ac.jp*

Manuscript received xx xx, 2017; revised xx xx, 2017.

Fig. 1. Robotic for Disaster Scenarios.

environments. Thanks to the previous works in distributed wireless sensor networks and robotic [4] [5] [6] [7] [8], we can focus on the perception method in this paper, and will detail it in the following sections. The main contributions of our work include:

- We propose a scene understanding method requiring only depth information instead of unstable RGB data, which gives robotic the ability to work in extreme conditions. We use two different 3D encodings for object proposal and classification, respectively, to further improve the performance of scene understanding.
- We design a view-invariant module, which can extract similar features from different views on a same object and adapt robotic for disordered environments.
- We work out a multi-task learning method. In addition, we also design a fast optimization method to accelerate the optimization process of complex loss functions.

2 Related Works

2.1 Object Detection and Localization

This is an old topic emerging with the boom of 2D image processing, detection, segmentation and classification. It is a prerequisite of the object classification in images and scenes, where lots of objects exist and mix together in a complicated way. A simple and effective method to solve this problem is the exhaustive search, i.e. searching all possible locations with all possible scales, which is called sliding window. Dalal and Triggs [9] use a conventional non-maximum suppression to run with detection windows to detect object instances. Harzallah et al. [10] apply SVM for each sliding window to select a few candidate regions, and use a score function on these regions to give out the final results.

However, the traditional methods working well in 2D images cannot be directly adopted in 3D images. Much more work needs to be done for 3D object proposal applications. Gupta et al. [11] extends Multiscale Combinatorial Grouping (MCG) framework [12] to 3D, and use RGB-D contours to calculate object candidates using features of depth and color images. Song and Xiao [13] present a CNN model taking a 3D volumetric scene as input to predict the object boundaries.

We desire to develop a CNN based object detection and localization network, and adopt a single model for both two tasks. We mainly focus on a multi-task loss function, which makes the proposed method significantly different from existing ones.

2.2 Object Classification

With the rise of artificial intelligence, object classification has become a booming research area in recent years. Lots of encouraging approaches [14] have been proposed in this field, especially for 2D images. Krizhevsky et al. [15] design a deep CNN model and achieve amazing classification accuracy. They use a novel regularization method called "dropout" to reduce overfitting which frequently happens in deep learning. Their model is proved very effective and becomes one pioneer in CNN based object classification methods. Szegedy et al. [16] present a large CNN architecture and set a new state-of-the-art for object classification. They introduce a carefully crafted design which can increase the size of network and keep the computational budget constant.

Again, traditional 2D methods meet frustrations in 3D world, due to the unique structure and characteristics of 3D data. Many researchers try to address this problem from various angles. Socher et al. [17] adopt two CNN models to extract low-level features from RGB and depth data separately, and an RNN model to combine these two parts and learn high-order features. The research of Shi et al. [18] is the most similar work with our method. They also achieve a rotation invariant representations for 3D shape. The difference is they adopt a row-wise max-pooling layer to change the low layer features extracted by the convolutional layers, while we implement view-invariant during learning process using specially-designed loss functions. The view-invariant ability improves the precision and adaptability of our method in some extreme conditions.

2.3 Robotic

Robotic is a huge topic consisting of many areas. As the paper is focused on robotic vision and proposes a 3D understanding method for mobile robot in disaster scenarios, only some closely related works are reviewed in this section.

One of the key problem in robotic vision applications is the acquisition of 3D information, i.e. 3D model and surrounding environment. Fortunately, many efficient approaches have been proposed, e.g., Aleotti et al. [19] present a method to automatically reconstruct the unknown objects. Vasquez-Gomez et al. [20] introduce a view planning method for a mobile robot. In [21], we also propose a robotic view planning and 3D reconstruction method, which greatly improve the efficiency and quality in acquiring 3D information. All these efforts contribute to increasing the quantity of available 3D models and accelerating the speed of robots when exploring unknown world.

2.4 Distributed Deep Learning

Although deep learning has achieved huge success in the last few years, one major drawback is that the training process remains very slow, especially for large datasets. Distributed computing can serve as a good solution for this problem.

Chung et al. [22] introduce their experiences on applying parallel technologies to train deep networks. They adopt a data-parallel Hessian-free optimization algorithm with the IBM Blue Gene/Q system. Das et al. [23] work out a

TABLE 1
Main Notations

Notation	Description
$\mathcal{X}$	Set of Input data
$\mathcal{Y}$	Set of ground-truth category labels $\mathcal{Y} = \{y_{x_1}, ..., y_{x_n}\}$
$\mathcal{T}$	Set of ground-truth 3D-box labels $\mathcal{T} = \{t_{x_1}, t_{x_2}, ..., t_{x_n}\}$
θ	A layer in proposed network
c_θ	Neuron number of layer θ
$\boldsymbol{h_\theta(x_i)}$	Output of layer θ
$\mathcal{W}$	Weights of proposed network, $\mathcal{W} = \{\boldsymbol{W_1}, \boldsymbol{W_2}, ..., \boldsymbol{W_\eta}\}$
$\boldsymbol{W_\theta}$	Weight matrix of layer θ, $\boldsymbol{W_\theta} = (\boldsymbol{w_\theta^1}, \boldsymbol{w_\theta^2}, ..., \boldsymbol{w_\theta^m})$
$\mathcal{B}$	Biases of proposed network, $\mathcal{B} = \{\boldsymbol{B_1}, \boldsymbol{B_2}, ..., \boldsymbol{B_\eta}\}$
$\boldsymbol{B_\theta}$	Bias matrix of layer θ, $\boldsymbol{B_\theta} = (\boldsymbol{b_\theta^1}, \boldsymbol{b_\theta^2}, ..., \boldsymbol{b_\theta^m})$
$J(\mathcal{W}, \mathcal{B})$	Loss function of proposed network
λ	Term weights used in functions
p	Parameter of Minkowski distance

distributed synchronous Stochastic Gradient Descent (SGD) algorithm. They also analyze the scaling and optimal design points for different networks. Different from the existing approaches, the adopted distributed method in the paper is specially designed for the robotic scene understanding. It can increase the batch size while keeping the recognition precision and communication costs. We will detail it in Section 3.4.

3 System Overview

3.1 Notation and Problem Definition

We use boldface uppercase letters to denote matrix, e.g. $\boldsymbol{W_\theta}$, lowercase letters for vectors, e.g. $\boldsymbol{w_\theta^{(m)}}$, and calligraphic-font letters to represent sets, e.g. $\mathcal{W}$. Notations used in the paper are listed in Table 1.

Given a collection of 3D images $\mathcal{X}$, the goal of scene understanding can be described as follows. For each input image x_i, the proposed method should detect and localize as many as possible objects $\mathcal{O}$, and classify them with correct labels $\mathcal{L}$.

3.2 System Framework

Fig. 2 presents an overview of the proposed method. For better understanding, a brief introduction on the main concepts is given in this section.

For 3D scenes, two individual models are used to conduct the object proposal and classification task, respectively. The objective of object proposal is to find any potential objects in the images, and, if any, localize them with precise locations and boundaries. This is an essential procedure for the following classification process, as classification model can only deal with a single object and is easily confused if there are multiple objects in the classification area. Therefore, the classification network will use the output of object proposal to clarify what the object is. Before the introduction of these two models, an explanation on the 3D data representations will be given, because input data must be encoded before it is used.

In addition, we enable the learning process with distributed ability, which will be introduced in Section 3.4.

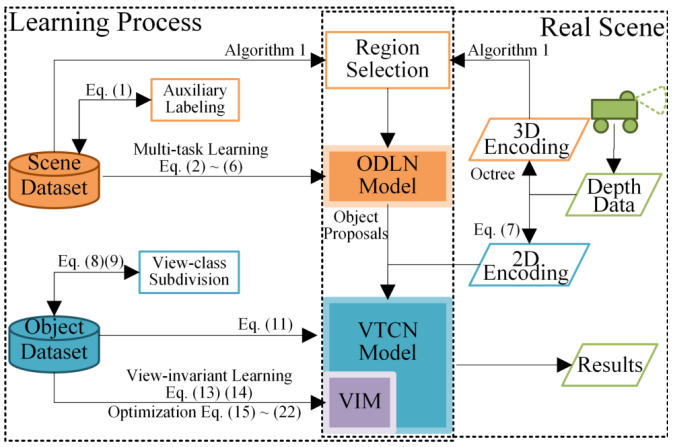


Fig. 2. Framework of the Proposed Method.

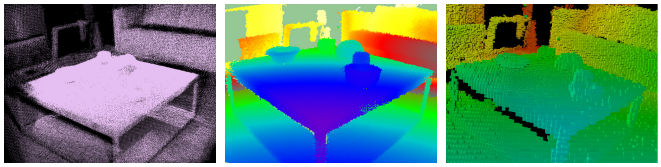


Fig. 3. 3D Scene Representation [24]. (a) Original point cloud. (b) 2D colored representation. (c) 3D volumetric representation (resolution = 0.02 m).

3.3 3D Data Encoding

The first challenge of 3D computer vision is the data encoding. Generally, the raw data captured by 3D sensors is in the form of point cloud, as shown in Fig. 3(a). This original scene comes from the RGB-D Object Dataset [24], which is also one of the datasets used in our learning and testing process. One common encoding method is to map the 3D point into RGB space, depending on the distance between the sensor and each point, as shown in Fig. 3(b), which is called the 2D colored representation. Another method is to transform the point cloud data into 3D volumetric representation, as shown in Fig. 3(c).

Currently, both methods have their advocates. However, according to [25], a CNN model trained with 2D-encoded data can obtain a significantly better performance in classification tasks than the one trained with 3D representation. This is mainly because that the advantages of 2D representation is very useful in classification tasks, e.g., the availability of well-trained network and the high resolution of 2D images. Therefore, input images are encoded to 2D representation in our classification model. However, in object detection and localization task, we find 3D volumetric representation is more preferable, as introduced in [26], owing to its excellent ability of preserving 3D spatial information. For this reason, 3D volumetric encoding is used in the detection and localization task. It is another difference between our job and existing approaches that we select different representations for different tasks, instead of using one same representation in all applications.

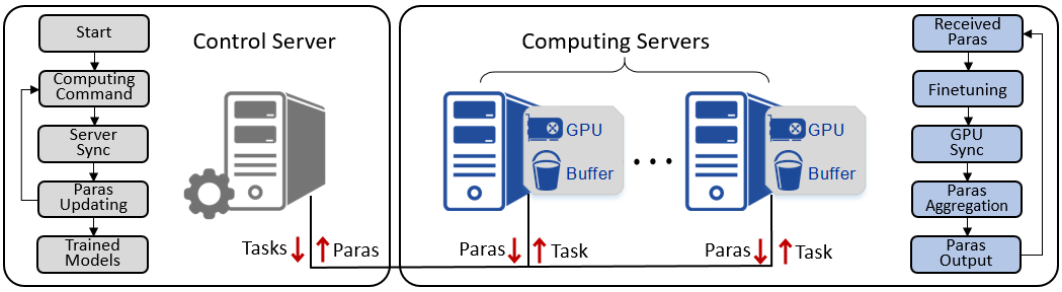


Fig. 4. The Distributed Design for the Deep Model Training.

3.4 Distributed Implementation

A major consideration when using deep learning systems is how to deal with the large data. In order to achieve a high precision, the deep models have to be trained with a huge number of labeled images, which will take significant time if only one computing node is used. To address this problem, the distributed computing is an obvious solution. Using distributed methods, the system can allocate the calculation tasks among multiple computing nodes, leading to higher efficiency and better load balance.

In the proposed 3D scene understanding system, the deep model, which is deployed in every mobile robot, should be updated with newly-labeled data. Therefore, fast on-line training is essential to adapt the robots into their current environments. Moreover, the uploaded data are located in different servers, it is more efficient to use distributed servers to process these data, than collecting them into one central server. Although many existing approaches can be applied to this scenario, we find it necessary to introduce our specially-designed distributed implementation, as it can keep the final precision while significantly increasing the speed of back-propagation.

A common way to address the network training problem is the mini-batch SGD, which can be expressed as

$$\boldsymbol{P}_{\mathbf{new}} \coloneqq \boldsymbol{P}_{\mathbf{old}} - \frac{\mathrm{lr}}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} \frac{\partial J}{\partial \boldsymbol{P}}$$

where $\boldsymbol{P}_{\mathbf{old}}$ is the existing parameters, $\boldsymbol{P}_{\mathbf{new}}$ is the updating parameters, lr is the learning rate, $\mathcal{B}$ is the batch, $|\mathcal{B}|$ is the batch size, x is the input data and J is the loss function. Mini-batch SGD is the common training method when using a standalone computing node. However, from the view of distributed computing, mini-batch is an inefficient way due to the frequently parameter synchronization, which results in large communication costs. Therefore, in our implementation, we adopt a large-batch SGD in order to fully utilize the advantages of distributed computing.

Large-batch SGD. Simply increasing the batch size may result in the degradation of the network precision. There are several points need to be noticed when using the distributed large-batch SGD. First, the learning rate lr should also be increased equally [27]. It is very important to match the training curve between large-batch and mini-batch SGD, which can improve the final precision of large-batch into the level of the mini-batch. Another point is the loss values should be normalized, which is also a key factor for the training result. Different from the common normalization approach, i.e., normalize each server's loss by the server number s, we find a more stable approach by normalizing the servers' loss by the total batch size $s|\mathcal{B}|$, where $|\mathcal{B}|$ represents the batch size [28]. An evaluation of the training precision is presented in the experiment section, in order to demonstrate the performance of our specially-designed large-batch SGD method.

Communication. The communication cost mainly consists of two parts, the intra-server communication and the inter-server communication. The former one is to sync the calculated parameters among the graphics processing units (GPUs) in a same server; the latter one is to transfer the parameters across different servers. Six main phases are included: (1) the parameters among local GPUs are collected; (2) the collected parameters are summed locally; (3) the results are transferred to each server; (4) the results are aggregated into the final parameters; (5) the final parameters are sent to each server; (6) then broadcasted to each GPU. Phase (1)(2)(6) belongs to the intra-server communication, which can be significantly accelerated using GPU kernels. Therefore, the main cost is resulted from the inter-server communication, including phase (3)(4)(5). Although the aforementioned communication procedure will bring some cost to the network training, a distributed system can give a huge boost to the computing efficiency. There have been several methods to implement this approach in a distributed environment. We have had many different attempts to improve the efficiency while keep the precision, and found this solution quite recently.

We implement this training system as an Apache Spark application, referring to the software configuration of [3]. There are one control server and some distributed computing servers. Each computing server is equipped with multiple GPUs. As shown in Fig. 4, the distributed framework has complete training abilities. At first, the control server starts the tasks by sending out the computing command. Then each computing server will fine-tune its received parameters, and sync them among local GPUs. After the parameters are aggregated, they will be transferred to the central server, following the server sync command from the control server. Then the aggregation of the collected parameters will be sent to each computing server as the updating parameters. After that, the computing servers will start another iteration of the fine-tuning. And when the loss value is converged, the control server will transfer the final parameters to the remote deep models, which are deployed in each mobile robot.

In addition, we conduct several evaluations in Section

6.6 to demonstrate its performance.

4 3D Object Proposal

4.1 ODLN Model

Object proposal is very important for the whole scene understanding process, because it can greatly decrease the recognition area and accelerate the classification network. A CNN-based Object Detection and Localization Network (ODLN) is proposed in this section to accurately propose potential object candidates for subsequent calculations. As previously mentioned, we desire to propose a scene understanding method without the requirement of RGB data which is full of uncertainty. As a result, it is very difficult to work out a selective search scheme like [29] and [13], in which color information is indispensable for similarity calculation. Therefore, we design a region selection method which is very similar with sliding window. By constantly selecting all possible regions in FOV, all objects could be included in at least one region, and get imported into ODLN for further calculation. We also adopt an auxiliary labeling method to help people mark a massive number of raw data and improve the practicability of supervised learning. The region selection algorithm and auxiliary labeling method are presented in Algorithm 1. Several hand-crafted features [30] are used to perform auxiliary labeling on initial dataset, these features are learnt with SVM. Fig. 6 shows the effects of the auxiliary labeling method and the proposed ODLN model. Although the hand-crafted features are outperformed by self-learning ODLN, it can save people lots of time by labeling the scene data automatically, especially when a large dataset is used for learning. The templates used for region selection include 10 cuboids in different sizes.

The network architecture of ODLN is presented in Fig. 5. There are two specially-designed modules and three ordinary convolutional layers behind the input layer, following by two individual sub-networks which will output detection and localization results respectively.

The input of ODLN is the volumetric representation of 3D objects in each region with their labels. The objective of ODLN is to judge whether there is a single object in the input region, and its exact boundaries and orientation.

Inspired by [31] and [16], we design the Scale-Insensitive Convolutional (SIC) module. There are three convolutional kernels with different size in one SIC module, which can extract the features from the input data in different scale. SIC module gives ODLN the ability to cope with 3D objects in various sizes.

Two sub-networks are behind the convolutional networks for detection and localization. The detection network outputs a value between $[0, 1]$ representing the possibility y that there is a single object in current region. The localization network is to predict the object's center point $\{x, y, z\}$, dimensions $\{\ell, w, h\}$ and Euler angles $\{\alpha, \beta, \gamma\}$, as shown in lower right corner of Fig. 5. Both sub-networks have two fully-connected layers for further analysis and abstracting on the features originate from the lower layers. Specially, we design a multi-task loss function to train two sub-networks at a same time, which is detailed in 4.2.

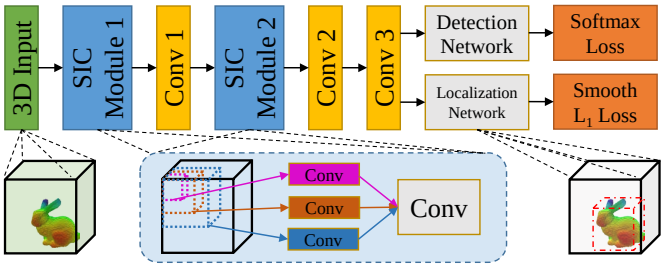


Fig. 5. Network Architecture of ODLN.

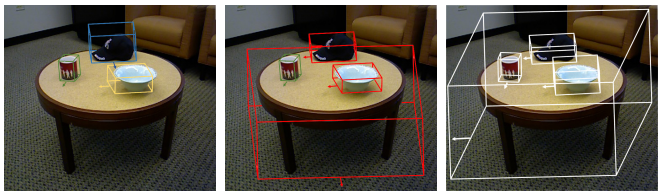


Fig. 6. 3D Scene [24] Labeling(resolution = 0.02 m). (a) Auxiliary labeling. (b) Manual amendment. (c) Actual proposal result using ODLN.

In short, 3D data in every possible region is imported into ODLN with their labels, which have the information including whether the region has an object and its locations. After forward propagation, ODLN can use the calculated values, their labels and the proposed loss function to adjust weights and optimize the network. During this process, ODLN can be gradually trained into a desirable network.

Algorithm 1: Region Selection and Object Proposal

Input: 3D Scene Data $\mathcal{S}$, Region Templates $\mathcal{T}$
Output: Object Proposals $\mathcal{P}$
proposal_list = null;
for $v \in \mathcal{S}$ **do** `/* Traversal of voxels. */`
  for $t \in \mathcal{T}$ **do** `/* Try all Templates. */`
    if *is_auxiliary_labeling* **then**
      `/* Perform auxiliary labeling */`
      Compute

$$\text{Score}(v) = \lambda_1\psi_{point_cloud} + \lambda_2\psi_{free_space} + \lambda_2\psi_{height_prior} + \lambda_2\psi_{height_contrast} \quad (1)$$

      if $\text{Score}(v) >$ threshold **then**
        Append v to proposal_list.
    else `/* Imported to ODLN Model. */`
      $\text{Score}(v), \text{Location}(v) = \text{ODLN} \leftarrow v$;
      if $\text{Score}(v) >$ threshold **then**
        Append $\text{Location}(v)$ to proposal_list.
return proposal_list;

4.2 Learning with Multi-task Loss Function

Given n input data $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ in ODLN, there are several different labels for each input data x_i, i.e., ground-truth labels $\mathcal{Y} = \{y_{x_1}, y_{x_2}, ..., y_{x_n}\}$ and 3d-box labels $\mathcal{T} =$

$\{t_{x_1}, t_{x_2}, ..., t_{x_n}\}$, where $t \in \{x, y, z, \ell, w, h, \alpha, \beta, \gamma\}$ standing for locations, dimensions and orientations respectively. In regard to the possible labels, $y \in \{y_1^*, y_2^*, ..., y_m^*\}$ and t should be real values. Specially, in the detection subnetwork of ODLN, $m = 2$ and $y \in \{0, 1\}$. $\boldsymbol{h_{\mathbf{cls}}(x_i)}$ is the output of Softmax classification layer, and $\boldsymbol{h_{\mathbf{loc}}(x_i)}$ is the one of Smooth L_1 Localization layer.

Inspired from the Multi-task Loss in [13] and [32], we design a loss function for ODLN as follows.

$$\begin{aligned} J(\mathcal{W}, \mathcal{B}, \mathcal{X}, \mathcal{Y}, \mathcal{T}) =& \lambda_1 J_{\text{cls}}(\mathcal{W}, \mathcal{B}, \mathcal{X}, \mathcal{Y}) \\ &+ \lambda_2 J_{\text{loc}}(\mathcal{W}, \mathcal{B}, \mathcal{X}, \mathcal{T}) + \lambda_3 J_{\text{dec}}(\mathcal{W}) \end{aligned} \tag{2}$$

where J_{cls}, J_{loc} and J_{dec} are sub-loss functions for different objectives, and λ_1, λ_2 and λ_3 are weights to custom the relative importance of each term.

J_{cls} is the softmax loss function for measuring the accuracy of the final detection result, i.e. whether a region owns one single object or not. It can be expressed as

$$\begin{aligned} & J_{\text{cls}}(\mathcal{W}, \mathcal{B}, \mathcal{X}, \mathcal{Y}) \\ &= -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{j=1}^{m} \mathbb{1}\{y_{x_i} = y_j^*\} \log \frac{e^{h_{\text{cls}}^{(j)}(x_i)}}{\sum_{\phi=1}^{m} e^{h_{\text{cls}}^{(\phi)}(x_i)}}\right] \\ &= -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{j=0}^{1} \mathbb{1}\{y_{x_i} = y_j^*\} \log p(y_{x_i} = y_j^* | x_i; \mathcal{W}; \mathcal{B})\right] \\ &= -\frac{1}{n}\left[\sum_{i=1}^{n} y_{x_i} \log(h_{\text{cls}}(x_i)) + (1 - y_{x_i}) \cdot \log(1 - h_{\text{cls}}(x_i))\right] \end{aligned} \tag{3}$$

where $h_{\text{cls}}^{(j)}(x_i)$ varies from 0 to 1, according to the possibilities that there is one single object in this region.

J_{loc} is utilized to predict the object's localization. Essentially, it is a regression of objects' boundary. According to the smooth L_1 loss in 2D bounding-box regression method of Fast R-CNN [32], J_{loc} is defined as

$$\begin{aligned} & J_{\text{loc}}(\mathcal{W}, \mathcal{B}, \mathcal{X}, \mathcal{T}) = \\ & \sum_{t \in \{x, y, z, \ell, w, h, \alpha, \beta, \gamma\}} \frac{1}{n} \sum_{i=1}^{n} \text{smooth}_{L_1}(\boldsymbol{h_{\mathbf{loc}}^{(t)}(x_i)} - \boldsymbol{t_{x_i}}) \end{aligned} \tag{4}$$

and

$$\text{smooth}_{L_1}(z) = \begin{cases} 0.5z^2, & |z| < 1 \\ |z| - 0.5, & |z| \geq 1 \end{cases} \tag{5}$$

where t represents the coordinates of the object's center point $\{x, y, z\}$, dimensions $\{\ell, w, h\}$ and Euler angles $\{\alpha, \beta, \gamma\}$.

$J_{\text{dec}}(\mathcal{W})$ is a weight decay term which penalizes large values of the parameters to decrease the magnitude of the weights $\mathcal{W}$ and help prevent over-fitting. It is computed by the following function.

$$J_{\text{dec}}(\mathcal{W}) = \| \mathcal{W} \|_2^2 \tag{6}$$

The proposed loss function takes object's detection and localization into account at a same time, which is a huge advantage compared to the common-used loss functions. As a result, the output of ODLN can be directly used as the input of object classification network. The performance of ODLN is evaluated in 6.1.

5 3D Object Classification

5.1 VTCN Model

A View-invariant 3D Classification network (VTCN) is proposed in this section. As mentioned above, 2D encoded image is adopted as the input of network. The depth value can be transformed to RGB data using the following formula.

$$\text{depth} \mapsto \underset{(0\sim360, 1, 255)}{(h, s, v)} \mapsto (r, g, b) \tag{7}$$

Therefore, VTCN has a similar structure with common 2D image classification CNN models, e.g. AlextNet [15], which is used as the prototype of the proposed VTCN model. To develop a classification model used in disordered and extreme scenarios, we desire to develop a view-invariant CNN model for feature extraction. We gain inspiration from [33] and proposed a new method which can be used with 3D data. A View-invariant (VI) Module consisting of two fully-connected layers is added between the 6th and 7th layer of AlexNet model. We also present a new object category subdivision method to give VTCN the ability to deal with multi-view images. With the well-learnt low-layer features brought by AlexNet, the high-layer features can get fine-tuned using 3D data in 2D representation. This learning process is conducted using specially-designed loss functions.

The class subdivision method and learning process will be detailed below, respectively.

5.2 View-subclass Definition

One of the most significant differences between VTCN and other methods is, one object category is not regarded as one single class. As is known to us, even a same object looks very different in different views. For example, it is very difficult to recognize a can if the viewpoint is directly below it. There would be even greater difficulties for robots with only depth information. Given these facts, we subdivide the existing object categories into several "view-subclasses", which will be the ground-truth label imported into VTCN model, as shown in Fig. 7.

So how to definite a view-sub-class? Interestingly, this is very similar to robotic view planning problem. In view planning methods, a next-best-view (NBV) is generated during each iteration of model reconstruction, then the depth sensor is moved to NBV to conduct the next scan. In the same way, several NBVs can be generated for each object category and each of them is used as a cluster center for one view-subclass. Fortunately, we have worked out an efficient robotic view planning method in [21], which can be directly used in VTCN. Because this algorithm is not the main topic in the paper, it will merely be given a brief introduction.

At first, NBV candidates are generated at all possible viewpoints in the whole working space, using the hierarchical generation and filtration algorithm proposed in our past work [21]. And they are evaluated using the score function presented below.

$$N(p) = \omega_1 \, \text{volume}(p) + \omega_2 \, \text{overlap}(p) \tag{8}$$

where $\text{volume}(p)$ represents the number of newly-observed voxels, $\text{overlap}(p)$ is the coincidence region between the

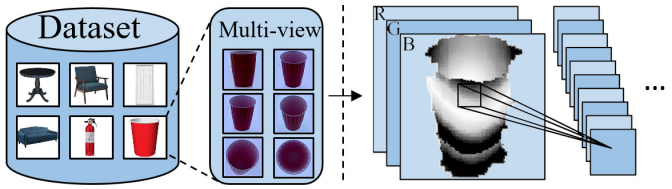


Fig. 7. View-subclass Division and the Input of VTCN.

new view and existing views. These two terms can control the number and orientation of the view-sub-class for each object category. $N(p)$ is the view point's score. And, the view with highest $N(p)$ score is selected as a new view-subclass.

After the sub-class centers are generated, they are extend to the whole working space. All of the viewpoints will be subjected a view-sub-class, depending on their similarities with the class centers. Given the points set $\mathcal{P}$ obtained in view v, the similarity function can be described as

$$\text{Similarity} = \frac{\mathcal{P}_v \bigcap \mathcal{P}_{\text{center}_\nu}}{\mathcal{P}_v \bigcup \mathcal{P}_{\text{center}_\nu}} \tag{9}$$

Then every view belongs to a center ν, which owns a view-sub-class, with the highest similarity score.

5.3 Learning View-invariant Features

Suppose there are n object view-subclasses, then the input data of VTCN can be defined as

$$\begin{cases} \mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_n\} \\ \mathcal{X}_i = \{x_i^{(1)}, x_i^{(2)}, ..., x_i^{(i_m)}\} \\ \Rightarrow \mathcal{X} = \left\{\{x_1^{(1)}, x_1^{(2)}, ..., x_1^{(1_m)}\}, ..., \{x_n^{(1)}, ..., x_n^{(n_m)}\}\right\} \end{cases} \tag{10}$$

where $\mathcal{X}$ is the set of all input samples, and $\mathcal{X}_i$ is the set of all the views of one single view-subclass. Each view-subclass has i_m elements. Then the total number of input data $\text{SIZE}(\mathcal{X}) = \sum_i^n i_m$. The labels of input images can be represented by $\mathcal{Y} = \{y_{\mathcal{X}_1}, y_{\mathcal{X}_2}, ..., y_{\mathcal{X}_n}\}$.

Like Eq. (2), the loss function can be written as

$$\begin{aligned} J(\mathcal{W}, \mathcal{B}, \{\mathcal{X}_i\}, \mathcal{Y}) = \quad & \lambda_1 J_{\text{cls}}(\mathcal{W}, \mathcal{B}, \{\mathcal{X}_i\}, \mathcal{Y}) \\ & + \lambda_2 J_{\text{view}}(\mathcal{W}, \mathcal{B}, \{\mathcal{X}_i\}) \\ & + \lambda_3 J_{\text{dec}}(\mathcal{W}) \qquad (0 \le i \le n) \end{aligned} \tag{11}$$

The difference is that Eq. (11) needs $\{\mathcal{X}_i\}$ as the view-subclass information to learn view-invariant features.

Similarly, $J_{\text{cls}}(\mathcal{W}, \mathcal{B}, \{\mathcal{X}_i\}, \mathcal{Y})$ can be described as

$$\begin{aligned} & J_{\text{cls}}(\mathcal{W}, \mathcal{B}, \{\mathcal{X}_i\}, \mathcal{Y}) \\ & = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{r=1}^{i_m}\sum_{j=1}^{c_{cls}} \mathbb{1}\{y_{x_i} = y_j^*\} \log \frac{e^{h_{\text{cls}}^{(j)}(x_i^{(r)})}}{\sum_{\phi=1}^{m} e^{h_{\text{cls}}^{(\phi)}(x_i^{(r)})}}\right] \\ & = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{r=1}^{i_m}\sum_{j=1}^{c_{cls}} \mathbb{1}\{y_x = y_j^*\} \log p(y_x = y_j^* | x_i^{(r)}; \mathcal{W}; \mathcal{B})\right] \end{aligned} \tag{12}$$

where c_{cls} is the neuron number of final layer, i.e., the total number of output features. And, $J_{\text{dec}}(\mathcal{W})$ is same with Eq. (6).

Then, the key problem is how to design a view-clustering function to give VTCN the ability to extract similar features from different views of a same object. Therefore, the differences in a view-sub-class $\mathcal{X}_i$ should be calculated and deteriorated during the learning process. Based on these assumptions, we define

$$\begin{aligned} & J_{\text{view}}(\mathcal{W}, \mathcal{B}, \{\mathcal{X}_i\}) \\ & = \frac{1}{pn}\left[\sum_{i=1}^{n}\sum_{r=1}^{i_m} \parallel \boldsymbol{h_{\text{vim}}}(\boldsymbol{x_i^{(r)}}) - \overline{\boldsymbol{h_{\text{vim}}}(\boldsymbol{\mathcal{X}_i})} \parallel_p^p\right] \\ & = \frac{1}{pn}\left[\sum_{i=1}^{n}\sum_{r=1}^{i_m}\sum_{k=1}^{c_{cls}} (h_{\text{vim}}^{(k)}(x_i^{(r)}) - \overline{h_{\text{vim}}^{(k)}(\mathcal{X}_i)})^p\right] \end{aligned} \tag{13}$$

and

$$\overline{h_{\text{vim}}^{(k)}(\mathcal{X}_i)} = \frac{1}{i_m}\sum_{r=1}^{i_m} h_{\text{vim}}^{(k)}(x_i^{(r)}) \tag{14}$$

where h_{vim} is the output of VI Module.

So the main objective is to optimize the following function

$$(\hat{\mathcal{W}}, \hat{\mathcal{B}}) = \arg\min_{\mathcal{W},\mathcal{B}} J(\mathcal{W}, \mathcal{B}, \{\mathcal{X}_i\}, \mathcal{Y}) \tag{15}$$

It is extremely difficult to perform this function on the entire dataset. Therefore, a batch Stochastic Gradient Descent (SGD) strategy is adopted in network learning, i.e. perform the optimization on a batch sampled from the whole dataset in each iteration. Although SGD has been proved successful on the functions like J_{cls} and J_{dec}, some more work is needed before it can be applied to the newly proposed J_{view} function. The stochastic and back-propagation method of J_{view} should be figured out to improve the computational efficiency.

Define bt is one batch and

$$\boldsymbol{H_{\theta,\text{bt}}} = \begin{pmatrix} \boldsymbol{h_\theta(x_1)} \\ \boldsymbol{h_\theta(x_2)} \\ \vdots \\ \boldsymbol{h_\theta(x_{n_b})} \end{pmatrix} \tag{16}$$

is a matrix consisting of all the outputs of layer θ in a batch. So the right-hand side of Eq. (13) can be expressed as

$$J_{\text{view}}(\mathcal{W}, \mathcal{B}, \mathcal{X}_{\text{bt}}) = \frac{\parallel \boldsymbol{H_{\text{vim,bt}}} - \overline{\boldsymbol{h_{\text{vim}}}(\mathcal{X}_{\text{bt}})}\mathbb{1}_{1\times n_b} \parallel_p^p}{pn_b} \tag{17}$$

where

$$\boldsymbol{H_{\text{vim,bt}}} = \boldsymbol{W_{\text{vim}}}\boldsymbol{H_{\text{vim-1,bt}}} + \boldsymbol{B_{\text{vim}}}\mathbb{1}_{1\times n_b} \tag{18}$$

w.l.o.g., assume that the average of feature is consistent during one iteration, and define $\mathbf{AVG} = \overline{\boldsymbol{h_{\text{vim}}}(\mathcal{X}_{\text{bt}})}\mathbb{1}_{1\times n_b}$ Then

$$\begin{aligned} \nabla_{J_{\text{view}}} &= \begin{bmatrix} \frac{\partial J_{\text{view}}(\mathcal{W}, \mathcal{B}, \mathcal{X}_{\text{bt}})}{\partial \boldsymbol{W_{\text{vim}}}} \\ \frac{\partial J_{\text{view}}(\mathcal{W}, \mathcal{B}, \mathcal{X}_{\text{bt}})}{\partial \boldsymbol{B_{\text{vim}}}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{(\boldsymbol{H_{\text{vim,bt}}} - \mathbf{AVG})^{p-1}}{n_b}\frac{\partial \boldsymbol{H_{\text{vim,bt}}}}{\partial \boldsymbol{W_{\text{vim}}}} \\ \frac{(\boldsymbol{H_{\text{vim,bt}}} - \mathbf{AVG})^{p-1}}{n_b}\frac{\partial \boldsymbol{H_{\text{vim,bt}}}}{\partial \boldsymbol{B_{\text{vim}}}} \end{bmatrix} \end{aligned} \tag{19}$$

Finally, the derivative of weights $\boldsymbol{W}$ and biases $\boldsymbol{B}$ of the last layer in VI Module can be expressed as

$$\frac{\partial J_{\text{view}}(\mathcal{W},\mathcal{B},\mathcal{X}_{\text{bt}})}{\partial \boldsymbol{W}_{\mathbf{vim}}} = \begin{cases} \dfrac{\text{sign}(\boldsymbol{H}_{\mathbf{vim,bt}} - \mathbf{AVG})}{n_b}\boldsymbol{H}_{\mathbf{vim\text{-}1,bt}} & ,p=1 \\ \dfrac{(\boldsymbol{H}_{\mathbf{vim,bt}} - \mathbf{AVG})^{p-1}}{n_b}\boldsymbol{H}_{\mathbf{vim\text{-}1,bt}} & ,otherwise \end{cases} \tag{20}$$

and

$$\frac{\partial J_{\text{view}}(\mathcal{W},\mathcal{B},\mathcal{X}_{\text{bt}})}{\partial \boldsymbol{B}_{\mathbf{vim}}} = \begin{cases} \dfrac{\text{sign}(\boldsymbol{H}_{\mathbf{vim,bt}} - \mathbf{AVG})}{n_b} & ,p=1 \\ \dfrac{(\boldsymbol{H}_{\mathbf{vim,bt}} - \mathbf{AVG})^{p-1}}{n_b} & ,otherwise \end{cases} \tag{21}$$

The weights and biases are updated as

$$\begin{aligned} \boldsymbol{W}_{\mathbf{vim}} &\coloneqq \boldsymbol{W}_{\mathbf{vim}} - \text{lr}\,\frac{\partial J_{\text{view}}}{\partial \boldsymbol{W}_{\mathbf{vim}}} \\ \boldsymbol{B}_{\mathbf{vim}} &\coloneqq \boldsymbol{B}_{\mathbf{vim}} - \text{lr}\,\frac{\partial J_{\text{view}}}{\partial \boldsymbol{B}_{\mathbf{vim}}} \end{aligned} \tag{22}$$

where lr is the learning rate. Then the stochastic is back-propagated to the lower layers. After a number of iterations, the loss function can convergence and give VTCN the ability to extract view-invariant features.

In addition, $\| \boldsymbol{h}_{\mathbf{vim}}(\boldsymbol{x}_i^{(r)}) - \overline{\boldsymbol{h}_{\mathbf{vim}}(\boldsymbol{\mathcal{X}_i})} \|_p^p$ is the Minkowski distance of order p between $\boldsymbol{h}_{\mathbf{vim}}(\boldsymbol{x}_i^{(r)})$ and $\overline{\boldsymbol{h}_{\mathbf{vim}}(\boldsymbol{\mathcal{X}_i})}$. When $p = 1$, Minkowski distance degenerates into Manhattan distance, and when $p = 2$, it becomes Euclidean distance. For better learning efficiency, the p value should be carefully selected. Therefore, an experiment is conducted in 6.3 to explore the relationship between p value and the training performance. In addition, a mathematical proof is presented in the appendix to find a solution which can guarantee the adaptability of our system in different datasets.

6 Performance Evaluation

To fully reach the potentiality of the proposed networks, several existing datasets combined with our newly-scanned 3D data are utilized in the training and testing process. Our networks are complicated enough to contain more than one dataset without under-fitting. The adopted datasets include *NYU Dataset v2* [34], *A large dataset of object scans* [35], *RGB-D Object Dataset* [24], *Bigbird dataset* [36] and *SceneNN dataset* [37].

6.1 ODLN Performance

The proposed ODLN not only detect objects in images, but localize them with exact position, dimension and orientation. To compare ODLN with some existing approaches, the following formulas are adopted to evaluate the performance.

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{23}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \tag{24}$$

where TP, FP and FN mean true positive, false positive and false negative, respectively. It is common for the performance experiments of object proposal methods to present the relationship between Recall and Precision, because both Recall and Precision change with the settings of proposed object number.

In this experiment, all of the detection methods take 950 labeled 3D scenes, which contains 53 object categories, as the training material, and 50 raw point cloud data as the test set. Fig. 8 shows the comparison of ODLN and representative existing approaches, e.g. Sliding Shapes [38], SVM method [39] trained on Point Cloud & 3D Model Dataset and [39] trained on only Point Cloud. To demonstrate the adaptability for disaster scenarios, there are two versions of dataset prepared for our experiment: one normal dataset and its variant in which the brightness of RGB images is turned down and a part of depth information is removed manually. The first row in Fig. 8 presents the results of the normal dataset, and the second row presents the results of the variant dataset. Five categories are selected in the experiment, i.e., chair, bed, sofa, table and toilet.

As shown in Fig. 8, ODLN method performs better in the variant dataset. It is mainly because we desire to design a method not relying on instable RGB data from the beginning, and the proposed ODLN is not affected by the missing of RGB data. However, the incomplete depth data do give some impact on its performance. But still, it can outperform others in these difficult situations. Of course, ODLN does not show much advantage in the normal dataset, especially compared with Sliding Shapes method. But in our opinion, the normal dataset is too idealized and have little practical significance, considering the disaster scenarios.

6.2 View-invariance Test

A demonstrating test is conducted to validate the effect of VI Module in VTCN model. As shown in Fig. 9, nine views of bunny model are selected in this experiment. They belong to three different view-subclasses. Therefore, there will be nine features extracted from these views, and each set of three should be similar. The first row of Fig. 9 shows each scan view of the bunny model. The second row is the 2D representation of obtained 3D data, which is also the input of VTCN model. The third and fourth row present their features before and after the process of VI Module, respectively. The features are sampled out from the 4096 outputs of VTCN's last layer. It can be seen that, without VI Module, the features looks very different even in a same view-subclass, while with VI Module, views in one subclass become consistent and generate fairly alike features. To further figure out the differences, their variance is calculated using the following formula.

$$\sigma = (\boldsymbol{F_a} - \boldsymbol{F_b})^T(\boldsymbol{F_a} - \boldsymbol{F_b}) \tag{25}$$

The last row of Fig. 9 shows the variance curves of the features before and after the process of VI Module. Compared with the blue dotted line, the red one is smooth and steady, which gives a good proof to the proposed view-invariance method.

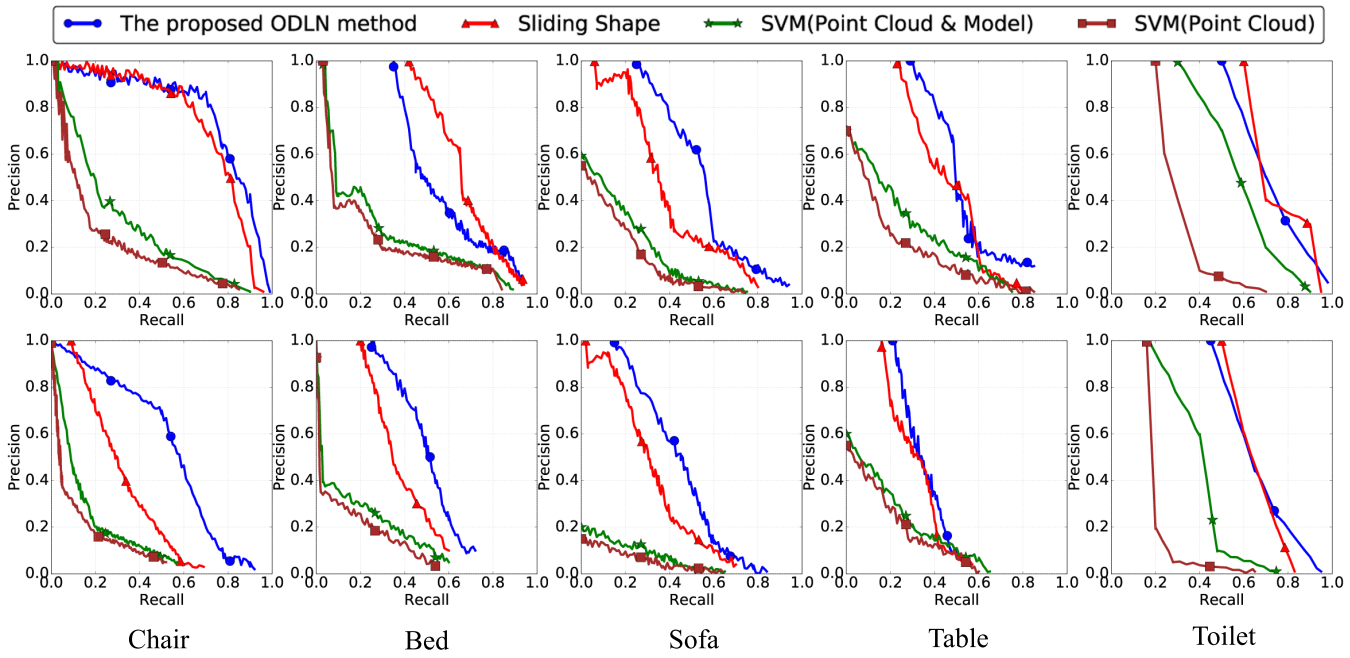


Fig. 8. Comparison of 3D Object Proposal Methods. Five object categories are adopted, i.e., chair, bed, sofa, table, and toilet. The first row shows the results conducted on a normal dataset, and the second gives the results of a more difficult variant.

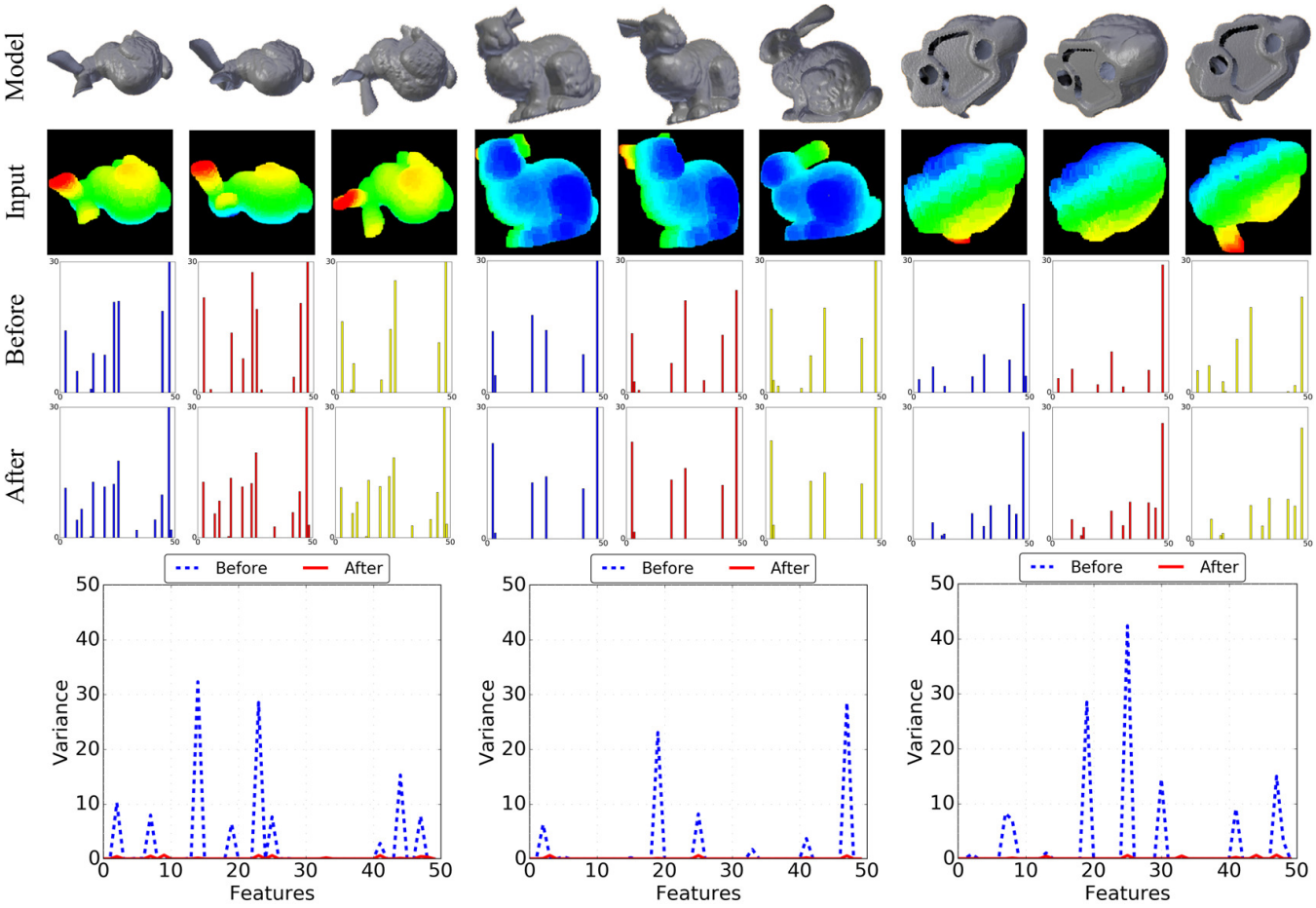


Fig. 9. Results of View-invariance Experiment. The first row is several views to scan the model; the second row is the 2D representation of the 3D data; the third row is 50 sampled features which are extracted without VI Module; the fourth row is the same 50 features extracted with VI Module; the last row gives the variance of each view-subclass.

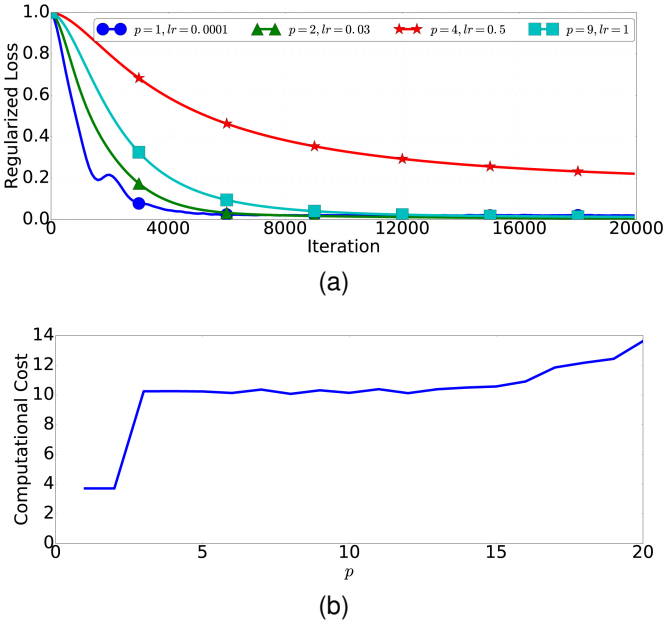


Fig. 10. Results of p Value Selection Experiment. (a) The regularized loss curves with different settings of p and learning rate. (b) The computational cost of different p values.

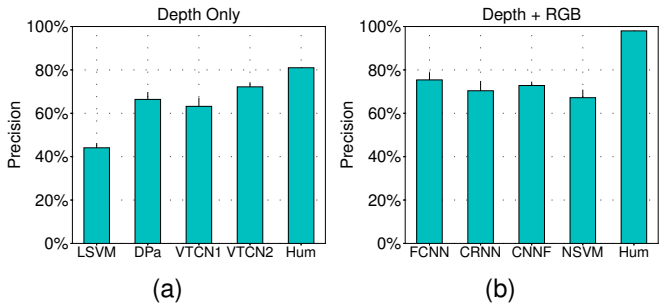


Fig. 11. Comparison Results of Classification Experiments. (a)Input data only contains depth information. (b) Input data contains both depth and RGB information.

6.3 p Value Selection

A set of experiment is designed to find the appropriate p value in the Minkowski distance for the loss function of VI Module. As discussed in 5.3, the setting of p value is very important for the training efficiency. To clarify the relationship between them, several tests are conducted using different settings. As expected, their results follow a specific rule. Fig. 10(a) presents four representative curves. A regularized loss is adopted in this experiment to measure the learning speed with different settings fairly.

$$\boldsymbol{J}_{i,\mathbf{reg}} = \left| \frac{\boldsymbol{J}_i^{(1)}}{J_i^{(1)}}, \frac{\boldsymbol{J}_i^{(2)}}{J_i^{(1)}}, \cdots \right| \tag{26}$$

where i represents the identifier number of each experiment with different p values. It can be seen, with the increase of p value, the learning process slows down significantly even with a larger learning rate. However, on the other hand, they are also more stable with a larger p when close to the convergent point.

Fig. 10(b) shows the relationship between computational cost and p values. The computational cost is defined as the complexity of calculation. Several unit operations are added in optimization algorithm to give out a uniform measure for estimated computational cost, whose value is defined in $0\sim50$. As shown in Fig. 10(b), the estimated cost increases with the p value, but not changes drastically as supposed. Therefore, the selection of a larger p is feasible because of the acceptable extra computational cost.

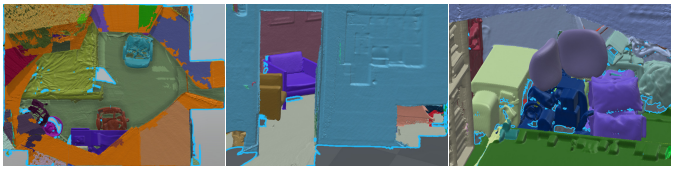
The results in this section are consistent with our theories and give us a helpful guide to select p values. A mathematical proof is presented in the appendix, in order to find the precise relationship between p value and training efficiency, and guarantee the adaptability of our system in various datasets.

6.4 Classification Performance of VTCN

A comparison experiment is performed to evaluate the classification performance of the proposed VTCN with several state-of-the-art methods. In this experiment, there are 53 categories of 3D labeled objects in the training and testing dataset. For each category, there are 100 instances in the training set, and 10 instances in the testing set. The results are shown in Fig. 11. Two set of experiments are designed. Although one same dataset is adopted in all experiments, one set is conducted without RGB information. The left figure presents the performance of depth-only methods, including Linear SVM [24], DPano [18], and the proposed VTCN method; the right one shows the RGBD methods, i.e., Fus-CNN [40], CNN-RNN [17], CNN-Features [41], and Kernel SVM [24]. Our VTCN method is tested without VI Module (VTCN1) and with VI Module (VTCN2), respectively, in order to present the performance difference between with and without the view-invariant ability. For better understanding, a manual classification test is also performed. It can be seen that our method improves a lot with the view-invariant ability of VI Module, and outperforms other depth only methods significantly. Although with similar view-invariant feature, VTCN can still outperform DPano method. However, different from our approach, DPano adopts a row-wise max-pooling layer to change the low layer features extracted by the convolutional layers, which needs a semi-panoramic view of the objects. This solution can lead to serious problems in some actual scenes, which will be discussed in Section 6.5. But in this test, we have given DPano the necessary views to perform the recognition. Still, its precision is lower than ours, because VTCN further refines the object category using the view-subclass algorithm. The performance of VTCN is slightly inferior to RGBD methods, however, we believe, the actual conditions of disaster scenarios do limit the acquisition of RGB data. It is worthwhile to leave out the unstable color data and work on the 3D information for a scene understanding method, which is more robust and adaptable.

6.5 Disaster Adaptability

Although the above evaluations have indicated the performance of four different approaches and clearly demonstrate the advantage of our method in common scene understanding tasks, we find it is necessary to conduct an additional experiment for the disaster scenarios, because the extreme conditions of the disaster scenes may be a huge challenge



(a) (b) (c)

Fig. 12. Simulated Disaster Scenarios for System Evaluation.

to the perception methods. In order to simulate the disaster scenarios, we work out a new dataset consisting of more than 300 scenes. These scenes are selected from the adopted datasets, which have been introduced in Section 6. All of these images are made highly disordered and broken, and only contains the depth information, as shown in Fig. 12, which is very similar to some dark, extreme disaster situations.

We adopt the complete scene understanding system, including the trained ODLN and VTCN presented in Section 6.1 and 6.4, without extra training process. The new dataset is merely used as the test data. The objective is to evaluate the system performance in simulated disaster scenes. The system will first use ODLN to find possible objects, and then use the VTCN to recognize them. The classification precisions are recorded as the final result. With VI Module, the proposed system can achieve 55.61% in this test, while without VI module, it is 47.71%. As comparison, we also combine ODLN, VTCN with several other approaches for the same test data. The ODLN&Linear SVM [24] is 30.39%, the ODLN&DPano [18] is 37.16%, the SVM [39]&VTCN(with VI module) is 18.33%. Due to the difficult situation, the results are generally not very high. However, the combination of VTCN, ODLN and VI module does bring a significant improvement to the precision, which can give a boost to the development of disaster robotic. In addition, since the system cannot obtain the complete depth information of the objects, DPano method has no sufficient view to perform panoramic recognition, therefore, its performance on this dataset is not very good, if compared to its precision in Section 6.4.

6.6 Distributing Efficiency

In order to evaluate the proposed distributed training method, we adopt the Amazon Elastic Compute Cloud (EC2) service as the test environment. We apply ten p2.8xlarge instances. Each of them has one Intel Xeon E5-2686v4 Processor and 8 NVIDIA K80 GPUs. All of the instances support GPUDirect technology, which enables the peer-to-peer GPU communication. The instances are connected by wired network. One instance is the control server, and others serve as the computing servers, as introduced in Section 3.4.

These servers are used for the training of our deep models, using the aforementioned datasets. We conduct the complete training process several times with two different batch sizes, i.e., 512 and 16k, to demonstrate the advantages of the proposed large-batch SGD over the traditional mini-batch SGD. The results are shown in Fig. 13. The X-coordinate is the adopted GPUs, and the Y-coordinate is the computing

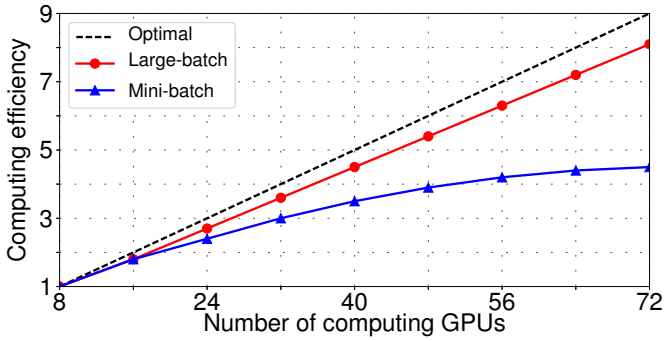


Fig. 13. Speedup of the distributed deep models. The dotted line is the optimal upper limit under full GPU utilization and zero communication cost. The red line is the proposed large-batch SGD; the blue line is a common mini-batch method.

efficiency, which represents the speedup compared to the standalone training. The dotted line is the optimal upper limit, which is under full GPU utilization and zero communication cost. The red line is the proposed large-batch SGD and the blue line is a common mini-batch method. It can be seen, that the large-batch SGD can approximately achieve a linear scaling, due to the significant decrease of the communication overhead. On the contrary, the mini-batch SGD suffers from an increasing communication cost and perform badly in highly distributed environments.

Another big difference is that our method can keep network precision with a large batch size. As comparison, we test the recognition performance of three trained networks. All of the three networks are trained using the same dataset, but with different training methods. One network is trained with the large-batch SGD, referring to the implementation introduced in Section 3.4, and its precision is 72.08%. The second network is trained with the normal large-batch SGD without any extra settings, and get a result of 65.79%. The third one is trained with the mini-batch SGD, scoring 72.65%. We can see that, compared with the third network, a large batch size does, to some extent, deteriorate the training effect. However, with the proposed large-batch configuration, this deterioration can be minimized and has little influence on the final precision.

6.7 Robotic Simulation

Robot Operating System (ROS) [42] is adopted as the simulation environment in this experiment. As shown in Fig. 14, a mobile robot moves along a path and takes three scans. The understanding results are generated by a system using the proposed method. To validate the feasibility of our method in dark scenes, only depth data is imported into the system. It can be seen, the proposed method is able to output correct labels for most objects. In Fig. 14(b), our system successfully recognizes the table with a confidence of 0.849, and the lamp with 0.424. It also generates several candidates with different confidences. For example, in Fig. 14(c), the system gives some labels to the desk in yellow zone including desk, table, cabinet, etc. Among them, desk is with the highest confidence, and becomes the final output label. The total precision is 91.0% with 10 runs. The results demonstrate that the proposed method is stable and robust,

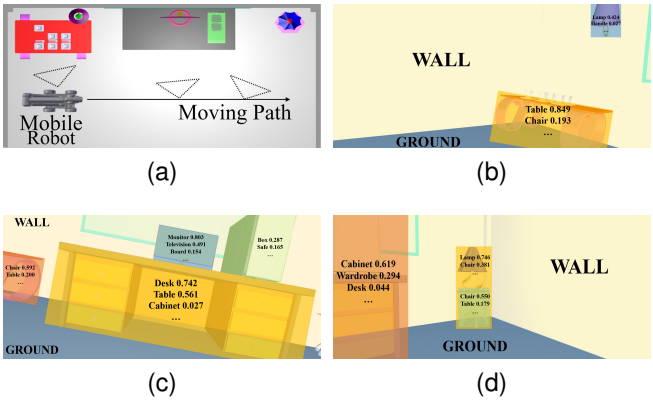


Fig. 14. Robotic Scene Understanding Simulation.

and can be well applied to complicated indoor scenes. In addition, we also use some existing approaches to perform the same task under the same condition. In fact, this is a similar test, compared to the one in Section 6.5, with more ordered environment and lower difficulty. The results are listed below. The ODLN&Linear SVM [24] is 35.62%, the ODLN&DPano [18] is 67.64%, the SVM [39]&VTCN(with VI module) is 41.81%.

7 Conclusion

A CNN-based robotic 3D scene understanding method is proposed in the paper. The most significant advantage of the method is its view-invariant ability and no requirement on RGB data, which are essential in disordered and extreme disaster scenarios. We propose several loss functions to perform multi-task learning process on the proposed models. And we also work out an optimization algorithm to improve the learning speed. The simulations demonstrate that the proposed method is effective, and outperforms many state-of-art approaches in several comparison experiments.

A limitation of the proposed system is that we still need labeled data in the training process. Therefore, in the future work, we will work on the automatic labeling problem to further decrease the system cost.

Appendix A
Proof of the Training Efficiency

Since Eq.(17) is the core component of the proposed loss functions, we should provide a formal proof in order to guarantee its training efficiency on different experiment datasets. The derivation of Eq.(17) has been presented in Section 5.3, however, the p value still has uncertain effect on the learning speed. Therefore, we will give a detailed proof on the relationship between the p value and the training efficiency. First, the following three lemmas are needed to prove the final theorem.

Lemma 1. *The convergence rate of the gradient descent method using different types of distances will follow*

$$ManhattanDistance > EuclideanDistance > ChebyshevDistance \tag{27}$$

Proof. This lemma can be expressed as the following statement. For two arbitrary points $P(x_0, y_0)$ and $Q(x_1, y_1)$, when $P \to Q$,

$$d_M(DP, DQ) \geq d_O(DP, DQ) \geq d_Q(DP, DQ) \tag{28}$$

where D is an arbitrary point.
Without loss of generality, we can assume D is the origin point $O(0, 0)$. Define $x_0 > y_0 > y_1$ and $x_0 > x_1$, then

$$\begin{aligned} d_M(OP, OQ) &= x_0 + y_0 - x_1 - y_1 \\ d_O(OP, OQ) &= \sqrt{x_0^2 + y_0^2} - \sqrt{x_1^2 + y_1^2} \\ d_Q(OP, OQ) &= x_0 - x_1 \end{aligned} \tag{29}$$

First, compare $d_M(OP, OQ)$ with $d_O(OP, OQ)$. We can start with the comparison between $\sqrt{x_0^2 + y_0^2} - \sqrt{x_1^2 + y_1^2}$ and $\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$. Then we can get

$$\begin{aligned} \sqrt{x_0^2 + y_0^2} - \sqrt{x_1^2 + y_1^2} &< \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2} \\ &< |x_0 - x_1| + |y_0 - y_1| \end{aligned} \tag{30}$$

Therefore, $d_M(OP, OQ) \geq d_O(OP, OQ)$.
Second, compare $d_O(OP, OQ)$ with $d_Q(OP, OQ)$

$$\frac{x_0 - x_1}{\sqrt{x_0^2 + y_0^2} - \sqrt{x_1^2 + y_1^2}} = \frac{\sqrt{x_0^2 + y_0^2} + \sqrt{x_1^2 + y_1^2}}{x_0 + x_1 + \frac{(y_0 - y_1)(y_0 + y_1)}{x_0 - x_1}} \tag{31}$$

Since $P \to Q$, we can get $x_0 \to x_1$ and $y_0 \to y_1$ with a same approaching speed. Therefore, $Eq.(31) \leq 1$. Therefore, $d_O(OP, OQ) \geq d_Q(OP, OQ)$ ■

Lemma 2. *There exists an intermediate point that enables us to express the Minkowski distance with the Euclidean distance. In other words, for two arbitrary points $P(x_0, y_0)$ and $Q(x_1, y_1)$, there exists a point $M(x_m, y_m)$ to make*

$$d_p(P, Q) = d_O(P, M) + d_O(M, Q) \tag{32}$$

Proof. Without loss of generality, we can assume Q is the origin point $O(0, 0)$. Then

$$d_p(O, P) = (x_0^p + y_0^p)^{\frac{1}{p}} \tag{33}$$

When $1 < p < 2$, we can get $x_0^p + y_0^p < (x_0 + y_0)^p$. Therefore, $d_p(O, P) < |x_0| + |y_0|$.
We need to prove that there always exists a point $M(x_m, y_0)$ which makes

$$x_m + y_0 = (x_0^p + y_0^p)^{\frac{1}{p}} \tag{34}$$

Because

$$(x_m + y_0)^p = \sum_{i=0}^{p} C_p^i x_m^i y_0^{p-i} \tag{35}$$

so it can be expressed as

$$x_1^p + \sum_{i=1}^{p-1} C_p^i x_m^i y_0^{p-i} = x_0^p \tag{36}$$

With the Intermediate Value Theorem, we can know Eq.(34) and (36) are valid.
When $p > 2$, we have $(x_0^p + y_0^p)^{\frac{1}{p}} < (x_0^2 + y_0^2)^{\frac{1}{2}}$.
Similarly, we can prove that there exists a point $M(x_0, y_m)$ to make $(x_0^p + y_0^p)^{\frac{1}{p}} = (x_0^2 + y_m^2)^{\frac{1}{2}}$. ■

Lemma 3. *Lemma 2 is valid for spaces with more than two dimensions. In other words, the distances can be projected into lower-dimension spaces.*

Proof. We have $x = (x_1, \cdots, x_n)^T$, $y = (y_1, \cdots, y_n)^T$ and $d_p(x, y) = [\sum_{i=1}^{n}(x_i - y_i)^p]^{\frac{1}{p}}$.
We need to prove that there exists a point $\xi = (\xi_1, \cdots, \xi_n)^T$ to make

$$d_p(x, y) = d_0(x, \xi) + d_o(y, \xi) \tag{37}$$

i.e.,

$$\Big[\sum_{i=1}^{n}(x_i - y_i)^p\Big]^{\frac{1}{p}} = \Big[\sum_{i=1}^{n}(x_i - \xi_i)^2\Big]^{\frac{1}{2}} + \Big[\sum_{i=1}^{n}(y_i - \xi_i)^2\Big]^{\frac{1}{2}} \tag{38}$$

which can be expressed as

$$\sum_{i=1}^{n}(x_i - y_i)^p = \left\{\Big[\sum_{i=1}^{n}(x_i - \xi_i)^2\Big]^{\frac{1}{2}} + \Big[\sum_{i=1}^{n}(y_i - \xi_i)^2\Big]^{\frac{1}{2}}\right\}^p \tag{39}$$

Similarly, using the Intermediate Value Theorem, we know that there exists $\xi = (\xi_1, \cdots, \xi_n)^T$ to make this equation true. ■

Lemma 4. *Lemma 1 is also valid for spaces with more than two dimensions.*

Proof. It can be deducted using Lemma 2 and 3. ■

Theorem 1. *For the proposed loss function $J_{\text{view}}(\mathcal{W}, \mathcal{B}, \{\mathcal{X}_i\})$, the training efficiency can be improved by decreasing the p value, i.e.,*

$$\nabla_{J^p_{\text{view}}} \geq \nabla_{J^{p+1}_{\text{view}}} \tag{40}$$

Proof. It can be deducted using Lemma 1, 2, 3 and 4. ■

According to the theorem, a smaller p value will increase the back-propagated stochastic and improve the optimization efficiency, which is also demonstrated in the experiment 6.3. On the other hand, a larger p value, resulting in a smaller stochastic, may also stable the learning process. Therefore, it is reasonable to set a small p in the early stage of training to get higher efficiency, and gradually increase p for stable convergence. The above proof guarantees that the proposed method can get a good learning speed with different datasets.

ACKNOWLEDGMENT

This work is partially supported by JSPS KAKENHI Grant Number JP16K00117, JP15K15976, and KDDI Foundation.

REFERENCES

[1] Z. Imani and H. Soltanizadeh, "Person reidentification using local pattern descriptors and anthropometric measures from videos of kinect sensor," *IEEE Sensors Journal*, vol. 16, no. 16, pp. 6227–6238, Aug 2016.

[2] Y. Kong, Z. Ding, J. Li, and Y. Fu, "Deeply learned view-invariant features for cross-view action recognition," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 3028–3037, June 2017.

[3] M. A. Alsheikh, D. Niyato, S. Lin, H. p. Tan, and Z. Han, "Mobile big data analytics using deep learning and apache spark," *IEEE Network*, vol. 30, no. 3, pp. 22–29, May 2016.

[4] H. Jiang, S. Zhang, G. Tan, and C. Wang, "Connectivity-based boundary extractionof large-scale 3d sensor networks: Algorithm and applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 4, pp. 908–918, 2014.

[5] M. M. Rashid, I. Gondal, and J. Kamruzzaman, "Share-frequent sensor patterns mining from wireless sensor network data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3471–3484, 2015.

[6] X. Hao, P. Jin, and L. Yue, "Efficient storage of multi-sensor object-tracking data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2881–2894, 2016.

[7] G. Zhan and W. Shi, "Lobot: Low-cost, self-contained localization of small-sized ground robotic vehicles," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 4, pp. 744–753, 2013.

[8] Y. Pei and M. W. Mutka, "Stars: Static relays for remote sensing in multirobot real-time search and monitoring," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 10, pp. 2079–2089, 2013.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[10] H. Harzallah, F. Jurie, and C. Schmid, "Combining efficient object localization and image classification," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 237–244.

[11] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *European Conference on Computer Vision*. Springer, 2014, pp. 345–360.

[12] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 328–335.

[13] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," *arXiv preprint arXiv:1511.02300*, 2015.

[14] X. Liu, L. Lu, Z. Shen, and K. Lu, "A novel face recognition algorithm via weighted kernel sparse representation," *Future Generation Computer Systems*, 2016.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[17] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3d object classification," in *Advances in Neural Information Processing Systems*, 2012, pp. 665–673.

[18] B. Shi, S. Bai, Z. Zhou, and X. Bai, "Deeppano: Deep panoramic representation for 3-d shape recognition," *IEEE Signal Processing Letters*, vol. 22, no. 12, pp. 2339–2343, 2015.

[19] J. Aleotti, D. L. Rizzini, R. Monica, and S. Caselli, "Global registration of mid-range 3d observations and short range next best views," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 3668–3675.

[20] J. I. Vasquez-Gomez, L. E. Sucar, and R. Murrieta-Cid, "View planning for 3d object reconstruction with a mobile manipulator robot," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 4227–4233.

[21] L. Li and N. Xiao, "Volumetric view planning for 3d reconstruction with multiple manipulators," *Industrial Robot: An International Journal*, vol. 42, no. 6, pp. 533–543, 2015.

[22] I. H. Chung, T. N. Sainath, B. Ramabhadran, M. Pichen, J. Gunnels, V. Austel, U. Chauhari, and B. Kingsbury, "Parallel deep neural network training for big data on blue gene/q," in *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2014, pp. 745–753.

[23] D. Das, S. Avancha, D. Mudigere, K. Vaidynathan, S. Sridharan, D. Kalamkar, B. Kaul, and P. Dubey, "Distributed deep learning using synchronous stochastic gradient descent," *arXiv preprint arXiv:1602.06709*, 2016.

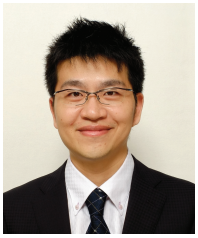
[24] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1817–1824.

[25] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 945–953.

[26] J. Liebelt and C. Schmid, "Multi-view object class detection with a 3d geometric model," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 1688–1695.
[27] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *arXiv preprint arXiv:1404.5997*, 2014.
[28] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.
[29] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.
[30] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," in *Advances in Neural Information Processing Systems*, 2015, pp. 424–432.
[31] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust object recognition with cortex-like mechanisms," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 3, pp. 411–426, 2007.
[32] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.
[33] G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 12, p. 7405, 2016.
[34] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.
[35] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun, "A large dataset of object scans," *arXiv:1602.02481*, 2016.
[36] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 509–516.
[37] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung, "Scenenn: A scene meshes dataset with annotations," in *International Conference on 3D Vision (3DV)*, 2016.
[38] S. Song and J. Xiao, "Sliding shapes for 3d object detection in depth images," in *European Conference on Computer Vision*. Springer, 2014, pp. 634–651.
[39] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 89–96.
[40] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 681–687.
[41] M. Schwarz, H. Schulz, and S. Behnke, "Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1329–1335.
[42] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

Liangzhi Li received the B.Sc and M.Eng degrees in Computer Science from South China University of Technology (SCUT), China, in 2012 and 2016, respectively. He is currently pursuing the Ph.D. degree in Electrical Engineering at Muroran Institute of Technology, Japan. His main fields of research interest include machine learning, big data, and robotics.

Kaoru Ota was born in Aizu-Wakamatsu, Japan. She received M.S. degree in Computer Science from Oklahoma State University, USA in 2008, B.S. and Ph.D. degrees in Computer Science and Engineering from The University of Aizu, Japan in 2006, 2012, respectively. She is currently an Assistant Professor with Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan. From March 2010 to March 2011, she was a visiting scholar at University of Waterloo, Canada. Also she was a Japan Society of the Promotion of Science (JSPS) research fellow with Kato-Nishiyama Lab at Graduate School of Information Sciences at Tohoku University, Japan from April 2012 to April 2013. Her research interests include Wireless Networks, Cloud Computing, and Cyber-physical Systems. Dr. Ota has received best paper awards from ICA3PP 2014, GPC 2015, IEEE DASC 2015, and IEEE VTC 2016-Fall. She is an editor of IEEE Communications Letters, Peer-to-Peer Networking and Applications (Springer), Ad Hoc & Sensor Wireless Networks, International Journal of Embedded Systems (Inderscience) and Smart Technologies for Emergency Response & Disaster Management (IGI Global), as well as a guest editor of ACM Transactions on Multimedia Computing, Communications and Applications (leading), IEEE Communications Magazine, etc. Also she was a guest editor of IEEE Wireless Communications (2015), IEICE Transactions on Information and Systems (2014), and Ad Hoc & Sensor Wireless Networks (Old City Publishing) (2014). She was a research scientist with A3 Foresight Program (2011-2016) funded by Japan Society for the Promotion of Sciences (JSPS), NSFC of China, and NRF of Korea.



Mianxiong Dong received B.S., M.S. and Ph.D. in Computer Science and Engineering from The University of Aizu, Japan. He is currently an Associate Professor in the Department of Information and Electronic Engineering at the Muroran Institute of Technology, Japan. Prior to joining Muroran-IT, he was a Researcher at the National Institute of Information and Communications Technology (NICT), Japan. He was a JSPS Research Fellow with School of Computer Science and Engineering, The University of Aizu, Japan and was a visiting scholar with BBCR group at University of Waterloo, Canada supported by JSPS Excellent Young Researcher Overseas Visit Program from April 2010 to August 2011. Dr. Dong was selected as a Foreigner Research Fellow (a total of 3 recipients all over Japan) by NEC C&C Foundation in 2011. His research interests include Wireless Networks, Cloud Computing, and Cyber-physical Systems. He has received best paper awards from IEEE HPCC 2008, IEEE ICESS 2008, ICA3PP 2014, GPC 2015, IEEE DASC 2015 and IEEE VTC 2016-Fall. Dr. Dong serves as an Editor for IEEE Communications Surveys and Tutorials, IEEE Network, IEEE Wireless Communications Letters, IEEE Cloud Computing, IEEE Access, and Cyber-Physical Systems (Taylor & Francis), as well as a leading guest editor for ACM Transactions on Multimedia Computing, Communications and Applications (TOMM), IEEE Transactions on Emerging Topics in Computing (TETC), IEEE Transactions on Computational Social Systems (TCSS), Peer-to-Peer Networking and Applications (Springer) and Sensors, as well as a guest editor for IEEE Access, Peer-to-Peer Networking and Applications (Springer), IEICE Transactions on Information and Systems, and International Journal of Distributed Sensor Networks. He has been serving as the Program Chair of IEEE SmartCity 2015 and Symposium Chair of IEEE GLOBECOM 2016, 2017. Dr. Dong was a research scientist with A3 Foresight Program (2011-2016) funded by Japan Society for the Promotion of Sciences (JSPS), NSFC of China, and NRF of Korea. He is the recipient of IEEE TCSC Early Career Award 2016.

Wuyunzhaola Borjigin received a B.S. degree and M.S. degree in School of Mathematical, Inner Mongolia University, China. She is currently working toward the Ph.D degree in the Emerging Networks and Systems Lab and Wireless Networks Lab, the Department of Information and Electronic Engineering, Muroran Institute of Technology, Muroran, Hokkaido, Japan, under the guidance of Prof. Mianxiong Dong. Her research interests include wireless networks, cloud computing, and cyber-physical systems.