Neurocomputing 323 (2019) 139–147

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

NormalNet: A voxel-based CNN for 3D object classification and retrieval

Cheng Wang[a], Ming Cheng[a,*], Ferdous Sohel[b], Mohammed Bennamoun[c], Jonathan Li[a,d]

[a] *Fujian Key Laboratory of Sensing and Computing for Smart City, School of Information Science and Engineering, Xiamen University, Xiamen, China*
[b] *Murdoch University, Perth, Australia*
[c] *The University of Western Australia, Perth, Australia*
[d] *Department of Geography and Environmental Management, University of Waterloo, Waterloo, Canada*

ARTICLE INFO

Article history:
Received 14 March 2018
Revised 29 August 2018
Accepted 28 September 2018
Available online 6 October 2018

Communicated by Liang Lin

Keywords:
3D object classification
3D object retrieval
Convolutional neural network
Network fusion

ABSTRACT

A common approach to tackle 3D object recognition tasks is to project 3D data to multiple 2D images. Projection only captures the outline of the object, and discards the internal information that may be crucial for the recognition. In this paper, we stay in 3D and concentrate on tapping the potential of 3D representations. We present NormalNet, a voxel-based convolutional neural network (CNN) designed for 3D object recognition. The network uses normal vectors of the object surfaces as input, which demonstrate stronger discrimination capability than binary voxels. We propose a reflection–convolution–concatenation (RCC) module to realize the conv layers, which extracts distinguishable features for 3D vision tasks while reducing the number of parameters significantly. We further improve the performance of NormalNet by combining two networks, which take normal vectors and voxels as input respectively. We carry out a series of experiments that validate the design of the network and achieve competitive performance in 3D object classification and retrieval tasks.

1. Introduction

As we live in a 3D world, recognition and analysis of 3D geometric models is an inevitable problem for computer vision research. With the emergence of large 3D repositories in the last several years [1–4], an extensive research on the classification, retrieval, and semantic labeling of 3D objects is becoming possible, and these areas have drawn great attention from researchers. Meanwhile, deep convolutional neural networks (CNNs) are introduced to 3D pattern recognition, and they are replicating the impressive success that they have achieved in the 2D field. Most of the recent systems which achieve state-of-the-art performance in 3D object classification on the ModelNet40 [4] benchmark are based on CNNs [5–10].

In this paper, we specifically focus on the classification and retrieval tasks of 3D objects obtained from CAD models and point clouds. A common approach to tackle these problems is to project 3D data to multiple 2D images, and a series of multiview-based 2D CNN architectures have been proposed [7,8,10,11]. Benefiting from the exhaustive 2D image classification research and the massive image databases that can be used to pre-train 2D CNNs (e.g. ImageNet [12]), these methods have outperformed their counterpart 3D voxel-based methods [4,13]. However, from the viewpoint of 3D data processing, projecting to 2D is a way of avoiding the issue instead of solving it. Projection only captures the outline of the object, and discards the internal information that may be important for the recognition of some specific categories. Fig. 1 gives an example. The middle of the figure shows a glassbox model. From outside, it is a simple cube, but in fact it has a complex internal structure. As the 3D data includes all original information, in this work, we concentrate on tapping the potential of 3D representations.

One obstacle to operate directly on 3D data is the computational and memory costs generated by the additional dimension. For example, images of 256×256 pixels are quite common for 2D image classification, while training a deep network on a 3D dataset with resolution 256^3 is computationally prohibitive. However, researchers have pointed out that the primary reason for the performance gap between 2D and 3D CNNs is the architecture of the network, not the input resolution [8]. In OctNet [14], the classification accuracy on ModelNet10 changes slightly between resolutions 32^3, 64^3, and 128^3, and even decreases with resolution 256^3. A 64^3 occupancy grid has the same size as a 512×512 image, which is totally affordable for current computers and GPUs. On the other hand, the amount of information in 3D data is not necessarily larger than 2D data. A 3D shape is defined on its surface, and

* Corresponding author.
E-mail address: chm99@xmu.edu.cn (M. Cheng).

https://doi.org/10.1016/j.neucom.2018.09.075
0925-2312/© 2018 Elsevier B.V. All rights reserved.

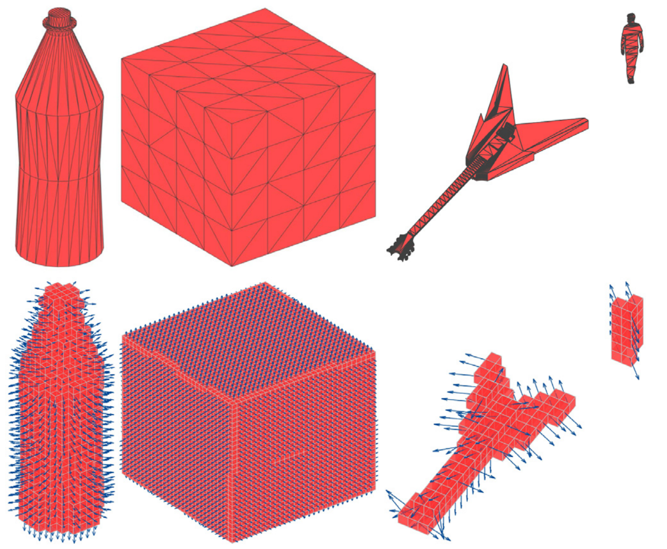

Fig. 1. Examples of CAD models from ModelNet40 and the corresponding (outward) normal vectors. From left to right: bottle, glassbox, and guitar. The models are voxelized with resolution 30^3, and the normal vectors of the surface voxels are calculated. The shown glassbox and guitar models have 10,942 and 119 surface voxels respectively, maximal and minimal in ModelNet40.

in a 3D occupancy grid each voxel is binary, whereas a 2D pixel has 256 gray levels. In this sense, a 3D classifier should be simpler than a 2D one.

In this paper we propose NormalNet, a voxel-based CNN with normal vectors as input. Intuitively, the performance of normal vector input should not be worse than voxel input, as the former contains more information (position and orientation of the model surface). In Section 4.1 we demonstrate that under the same network architecture, the normal vector input outperforms the voxel input consistently.

Furthermore, we improve the performance of NormalNet from three aspects: network architecture, data augmentation, and network fusion. We propose a reflection-convolution-concatenation (RCC) module to realize the conv layers in NormalNet, which generates feature maps by both conv kernels and their reflections. We use model rotation as a data augmentation method to increase training samples and reduce overfitting. We combine two networks, which take normal vectors and voxels as input respectively, and train them synchronously using the network fusion technique.

The key technical contributions of this paper are as follows: we propose to use normal vectors as input to a 3D CNN classifier, which outperform the voxel input. To the best of our knowledge, this is the first time that normal vectors are exploited in conjunction with volumetric CNN. We design an RCC module for the conv layer, which achieves higher classification accuracy than an ordinary conv layer with fewer parameters. We study the methods to combine multiple inputs, and further improve the performance in object classification and retrieval tasks.

The rest of this paper is organized as follows: In Section 2, we review current deep networks on 3D model. The proposed NormalNet is introduced in Section 3 including the network architecture, data augmentation method, and network fusion technique. Extensive experiments and results are given in Section 4. In Section 5, we summarize our conclusion.

2. Related work

A number of shape descriptors have been designed for 3D model analysis [15–17]. A comprehensive performance evaluation of 3D descriptors can be found in [18] where ten popular descriptors are compared in terms of descriptiveness, compactness, and robustness. Many of these descriptors are handcrafted towards specific tasks, and do not generalize well for shapes across a variety of classes with large variations. With the advancement in the field of deep learning, CNNs have been widely and successfully used on 2D and 3D data by the computer vision community. After trained on large dataset, CNNs can learn general purpose features that outperform handcrafted descriptors, and have achieved state-of-the-art results for various vision tasks.

Volumetric CNNs 3D CNNs have been used in video analysis [19,20], where time acts as the third dimension. The pioneer work in [4] made efforts to build deep learning models on 3D shapes directly. The authors trained a convolutional deep belief network (DBN) on a 30^3 voxel grid for shape classification, shape retrieval, and next-best-view prediction. They also released the large-scale CAD dataset ModelNet which boosts the research on 3D deep learning. A similar approach is VoxNet [13] which uses typical CNN architecture consisting of conv and fc layers. Following these works, Sedaghat et al. [21] studied the role of object orientation in 3D recognition. They trained a multi-task network which was forced to predict the pose of the object in addition to the class label. Volumetric CNNs are also part of the work in [8] where two distinct network architectures were proposed, i.e. CNN with auxiliary training by subvolume supervision and CNN with anisotropic probing kernels. These architectures work on volumetric occupancy grid, and can process different sources of 3D data, including CAD models, RGB-D point cloud, and LiDAR point cloud [22].

Further developments in volumetric CNNs include the applications of generative models [23], denoising auto-encoders [24], very deep discriminative architectures, and lightweight architecture. Brock et al. [5] trained networks with up to 45 layers and improved the ModelNet classification task by large margins. Zhi et al. [25] proposed LightNet to address the real-time 3D object recognition problem, which predicts class and orientation simultaneously and achieves the state-of-the-art 3D object recognition performance among shallow volumetric CNNs with the smallest number of training parameters.

Multiview CNNs The spatial resolution of volumetric CNNs is limited due to their computational and memory costs, and some works turn to relying on 2D CNNs fully or partly. In multi-view CNN (MVCNN) [8] 12 or 80 2D rendered images of a single object are aggregated by a view-pooling layer, and then passed to a CNN pre-trained on ImageNet to generate a compact shape descriptor. Johns et al. [7] applied CNN to generic multi-view recognition by decomposing an image sequence into a set of image pairs, classifying each pair, and weighting its contribution. Shi et al. [26] converted each 3D shape into panoramic views via a cylinder projection around its principle axis, then a CNN is used to learn the representations from these views. Sfikas et al. [9] also used the panoramic views consisting of 3-channel images, i.e., the spatial distribution map, the normals deviation map, and the magnitude of the normals deviation map gradient image. RotationNet [11] takes multi-view images of an object as input and jointly estimates its pose and object category. An ensemble of networks have been shown to boost the performance and this approach is adopted in FusionNet [6], which combines volumetric CNN and multiview CNN.

Point cloud CNNs Point cloud is becoming popular for representing 3D objects. It is an attractive approach to feed point clouds to deep networks directly since it does not need any transformation to the input data (i.e., voxelization or projection) and avoids loss of information. The difficulty is that point cloud is spatial irregular and permutation invariant, essentially different from rasterized data (pixel or voxel). Some networks have been proposed recently to challenge this difficulty. Kd-network [27] uses kd-tree to represent point clouds. The network mimics the operations in

CNNs with modifications to adapt to the input data structure. PointNet [28] directly takes point clouds as input, represented by three coordinates, and outputs class or part labels. A max pooling layer is applied before output to realize permutation invariance. PointNet++ [29] is later proposed by the same author which applies PointNet hierarchically for better capturing of local structures. Another two networks that consuming point coordinates are PointCNN [30] and SO-Net [31]. The former uses the so-called $\mathcal{X}$-Conv layers to extract features from input points. The latter models the spatial distribution of point cloud by building a self-organizing map (SOM), then performs hierarchical feature extraction on SOM nodes.

Although the idea of using coordinates as input is straightforward, the performance of current point cloud CNNs is still worse than multiview CNNs on 3D object recognition tasks.

3. Proposed method: NormalNet

NormalNet uses the unit normal vector as input, calculated for all surface voxels. In the process of voxelization, each patch of the CAD model is split iteratively until all vertices fall into the same voxel, then the value of the voxel is set to 1. A little extra operation can generate the normal vector at the voxel position by calculating the cross product of the two edges of the subpatch. For point cloud, the eigenvector corresponding to the minimum eigenvalue of the covariance matrix can be approximated as the normal vector. Fig. 1 shows several examples of CAD models from ModelNet40 and their corresponding normal vectors.

Compared with how to generate the normal vector, more attention should be paid to how to store it. If we use `float` type to store each coordinate component, a normal vector will take 12 bytes for storage, which is a huge burden on the memory. In practice we use one byte (`signed char` type) to store each coordinate, i.e., -128 represents -1 and $+127$ represents 0.9922, and the precision is $1/128 \approx 0.0078$. We find this precision is accurate enough for network training and inference.

After defining the input of NormalNet, we improve its performance from three aspects, which will be described in detail below. The feature generation method for object retrieval will also be presented.

3.1. Network architecture

NormalNet is a CNN consisting of conv and fc layers. We propose a new structure to realize the conv layer: the RCC module. In an RCC, the input of the conv layer combined with two of its reflection images are fed into the conv layer, and the conv results are concatenated along the channel dimension as the output of the layer (shown in Fig. 2). The consideration behind this idea is that if you can recognize an object, you can also recognize it from a mirror. As the feature extraction part in CNN, the conv layers should also have this capability. Thus, we send both the input and its reflections to the conv kernel, and force the conv kernel to extract features from multiple directions. This arrangement compels the conv kernel to be isotropic and to extract more distinguishable features for 3D vision tasks.

It is well known that parameter sharing is an important feature of CNN. The RCC module extends the idea of parameter sharing. The parameters are shared not only between different receptive fields, but also between the input and its reflections. This extension significantly reduces the number of parameters in the conv layers. In the case of the same number of output channels, the number of conv kernels is reduced to one-third.

A commonly used data augmentation technique in 3D CNN is data rotation (also used in this work) and randomly mirroring. The idea looks similar to RCC, but in fact there is essential difference.

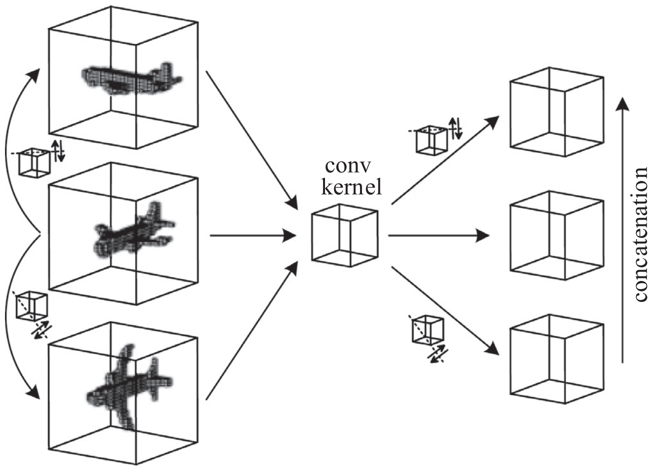

Fig. 2. In an RCC module, the input combined with two of its reflection images are fed into the conv layer. Considering that the input may have different sizes across dimensions, a second time of reflection is necessary before concatenating the conv results.

Data augmentation is applied to the input data, which are fed into the first conv layer only. Oppositely, RCC can be used in all conv layers. It reflects the output of the previous layer, and feeds the reflections and the original output into the next one.

When using the RCC module, there is a problem that needs to be considered: the size of the input. If the input has different sizes across dimensions, a second time of reflection has to be applied before concatenation. In practice we reflect the conv kernel instead of the input (shown in Fig. 3), and the results are exactly the same. In most applications, the kernel size and the stride are the same in different dimensions, and the conv results can be concatenated directly.

We define two types of RCC, denoted by RCC-I and RCC-II respectively. RCC-I includes only one conv layer, while RCC-II includes an additional conv layer with a kernel size of $1 \times 1 \times 1$. Convolution with kernel size 1 is common in 2D CNNs (e.g., in Inception Net [32]), which increases the feature transformation and nonlinearity with little cost. We apply this idea to RCC. Batch normalization [33] is used after each conv layer.

Let m, n_i, and n_o be the kernel size, number of input channels, and number of output channels, the numbers of trainable parameters in vanilla conv layer, RCC-I, and RCC-II are $m^3 n_i n_o + n_o$, $\frac{1}{3} m^3 n_i n_o + n_o$, and $\frac{1}{3} m^3 n_i n_o + n_o^2 + 2n_o$ respectively.[1]

The architecture of NormalNet is shown in Fig. 4. The 3D object is voxelized with resolution 30^3, and the three coordinate components of the normal vectors serve as the three input channels. The numbers of convolution output channels n_1, n_2, n_3 and the number of hidden units in fc layer n_4 are alterable, and we test different combinations in the experiments. Classification losses are computed using softmax and cross entropy.

3.2. Data augmentation

The appearance of the voxelized models depends on the orientation, and we augment the training data with multiple rotations. Two forms of rotation augmentation are compared [10]. The first form includes 12 rotations. Assuming that all models in the dataset are upright oriented along the z-axis (ModelNet40 satisfies this requirement), we create 12 copies for each model, each rotated at 30° intervals around the z-axis. The second form includes 20 rotations, and the viewpoints are set at the 20 vertices of an

[1] Batchnorm scaling is not useful with ReLUs.

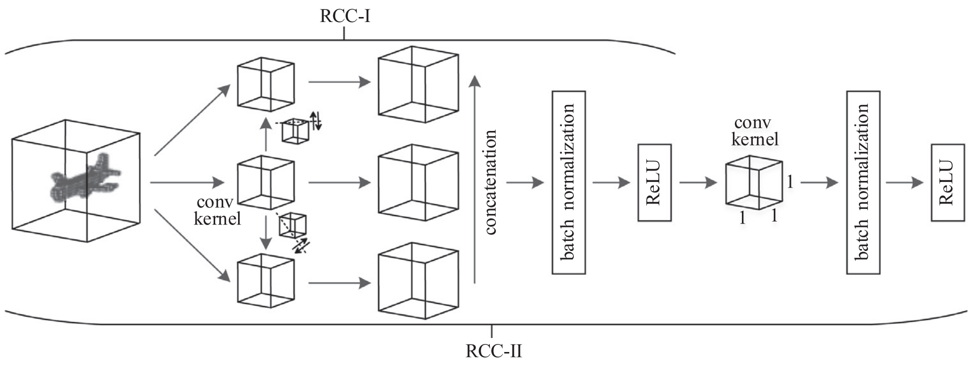

Fig. 3. Practical RCC module reflects the conv kernel instead of the input. In most applications, the conv kernel has the same size in different dimensions, thus the conv results can be concatenated along the channel dimension directly. We define two types of RCC, and RCC-II includes an additional conv layer with kernel size $1 \times 1 \times 1$.

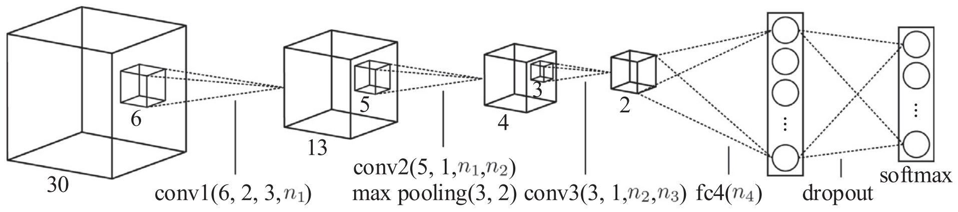

Fig. 4. NormalNet architecture, including three conv layers and one fc layer. The conv layer is defined by its kernel size, stride, number of input channels, and number of output channels. Both vanilla conv layer (with batch normalization) and RCC module are investigated.

icosahedron enclosing the model. At training time, the rotations of the same model are deemed to be different training samples. While during the test stage, all rotations of the same test sample are fed into the network in one batch, and the activations of the output layer are averaged.

3.3. Combination of multiple inputs

The fusion of multiple networks can achieve a better classification performance than separate ones. In [8] the best result is obtained by training an SVM over the concatenation of fc7 features from three networks. FusionNet [6] combines volumetric CNN and multiview CNN after the final fc layer. A linear combination of the fc outputs is used as the classification basis. In general, the fusion strategy in the present literature is relatively simple, i.e., combining the outputs of fc layers (concatenation or weighted average) as class scores, and the class corresponding to the highest score is declared to be the predicted class.

In this work, we combine normal vector input and voxel input for 3D object classification and retrieval using the network fusion technique. In the context of deep learning, network fusion can be realized with either hard or soft parameter sharing of hidden layers [34]. The former shares the hidden layers between all tasks while keeping several task-specific output layers, which greatly reduces the risk of overfitting. In the latter case, each task has its own model with its own parameters, while the distance between the parameters is regularized to encourage the parameters to be similar [35].

We use two networks denoted by N-net and V-net, which take normal vectors and voxels as input respectively. The structure of the two networks is almost identical except for the number of input channels. Two network fusion strategies are investigated (shown in Fig. 5). The first one is hard parameter sharing, which can cover one or more layers among the conv and fc layers. The second one is cross-stitch networks (CSNs) [36], in which the input of each layer is the linear combination of the activations of the previous layers from multiple networks. The combination may be layer-wise or channel-wise, and the coefficients are trainable.

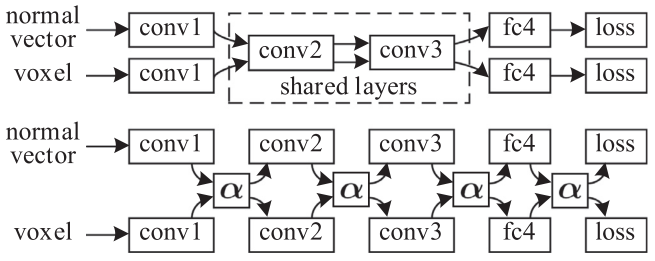

Fig. 5. Two network fusion strategies. Top: hard parameter sharing. The sharing can cover one or more layers from conv2, conv3, and fc4. Bottom: cross-stitch networks.

The improvement of the performance can be expected only when each network provides distinct information about the object. In our realization, the inputs of the two networks are from the same object, but they can be different rotations. Furthermore, we use anisotropic rescaling [27] as a data augmentation method for voxel input. This method is not suitable for normal vector input, since it changes the direction of the normal vector. The softmax outputs of the two networks are averaged and the index of the largest value indicates the class of the object. We also tried averaging the outputs of fc layers before softmax and we found the change of the performance is insignificant.

3.4. Feature generation for object retrieval

The features extracted by deep learning can be used for object retrieval as well. In this work, the activations of the fc layer of NormalNet for each input 3D model is deemed as the descriptor of that object. To perform the retrieval task, a 3D model descriptor is compared against the rest of the 3D model descriptors using the ℓ_2 distance metric.

Dimensionality reduction of the descriptor can yield a significant retrieval performance boost [8,10]. Let $\phi \in \mathbb{R}^d$ be the original descriptor, and we learn a projection matrix $W \in \mathbb{R}^{p \times d}$ with $p \ll d$ to generate a compact descriptor $W\phi \in \mathbb{R}^p$. The purpose of the projection is twofold: it improves the discrimination capability of the descriptors and dramatically reduces the dimensionality. We use the ratio between the intra-class distance and inter-class distance to evaluate the discriminative improvement of the projection. Suppose the dataset includes C classes, and class c has N_c samples. Let $\phi_i^{(c)}$ be the descriptor of sample i in class c, and $\mu^{(c)} = \frac{1}{N_c}\sum_{i=1}^{N_c}\phi_i^{(c)}$ be the center of class c. The optimal projection matrix should minimize the ratio:

$$W^* = \arg\min_W \frac{\sum_{c=1}^{C}\frac{1}{N_c}\sum_{i=1}^{N_c}\|W\phi_i^{(c)} - W\mu^{(c)}\|_2}{\sum_{1\le r<s\le C}\|W\mu^{(r)} - W\mu^{(s)}\|_2}. \quad (1)$$

We use the large-margin dimensionality reduction method in [37] to solve W^* iteratively. The method learns the projection matrix such that the squared Euclidean distance $\|W\phi_i - W\phi_j\|_2^2$ is smaller than a learnt threshold $b \in \mathbb{R}$ if samples i and j are in the same class, and larger otherwise. Furthermore, the method imposes a margin of at least one between the distance and the threshold, resulting in the constraints:

$$y_{ij}(b - \|W\phi_i - W\phi_j\|_2^2) > 1, \quad (2)$$

where $y_{ij} = 1$ iff i and j are in the same class, and $y_{ij} = -1$ otherwise. The solution is found using a stochastic sub-gradient method. At each iteration t, the algorithm samples a single pair of (i, j) and performs the following updates:

$$W_{t+1} = \begin{cases} W_t & \text{if (2) is satisfied} \\ W_t - \gamma y_{ij} W_t \Psi_{ij} & \text{otherwise} \end{cases} \quad (3)$$

$$b_{t+1} = \begin{cases} b_t & \text{if (2) is satisfied} \\ b_t + \gamma y_{ij} b_t & \text{otherwise} \end{cases} \quad (4)$$

where $\Psi_{ij} = (\phi_i - \phi_j)(\phi_i - \phi_j)^T$ is the outer product of the difference vectors, and γ is a constant learning rate.

4. Experiments and analysis

We now demonstrate the results of application of NormalNet to 3D object classification and retrieval. Our implementation of NormalNet uses the TensorFlow framework.

4.1. CAD shape classification

Dataset and training settings We use the popular ModelNet40 dataset for the 3D shape classification task. The dataset contains 40 object classes and consists of 9843 shapes for training and 2468 shapes for testing. Each shape is provided as a CAD model oriented in a canonical pose.

All normal vectors are precomputed and stored. When storing the normal vectors, the whole voxel grid is stored as well with one bit for each voxel. The 1-voxels are used as the indices of the normal vectors. If each model is voxelized with resolution 30^3 and has 20 rotations, the maximum and minimum numbers of surface voxels are 10,942 and 119 respectively, and the average number is 1795 (averaged over all 12,311 samples in the dataset and 20 rotations). The storage of all normal vectors needs 1.23 GB.

The network is trained end-to-end by the gradient descent optimizer. Conv kernels are initialized from a zero-mean truncated normal distribution with $\sigma = 0.1$. The loss function includes 0.001 times the ℓ_2 weight norm for regularization. The learning rate starts from 0.01 and is decreased by a factor of 2.5 each 15,000 epochs. Batch size is 64 and the probability that each element is kept in dropout is 0.75.

Voxel input vs. normal vector input We compare the performance of voxel input and normal vector input under the same network architecture in Table 1. The classification accuracy is averaged over sample and class respectively. The latter is lower because generally classes with fewer samples have lower classification accuracy.

Table 1
Classification accuracy on ModelNet40 (binary voxel input vs. normal vector input). The accuracy is averaged over sample and class respectively. The normal vector input outperforms the voxel input consistently.

network	voxel		normal vector	
	sample (%)	class (%)	sample (%)	class (%)
VoxNet [13]	84.2	82.3	84.9	83.1
ours-vanilla	86.7	84.1	87.8	85.8
ours-RCC-II	88.7	86.0	90.1	88.0

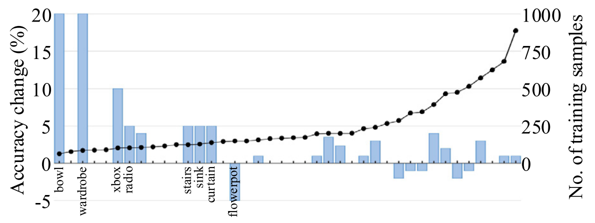

Fig. 6. Change of classification accuracy between normal vector input and voxel input per class (blue bars). The classes are sorted by the number of training samples (black line). Classes with accuracy change $\ge$ 5% are labeled. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Classification accuracy on ModelNet40 under various network configurations.

conv	(n_1, n_2, n_3, n_4)	#trainable	#rotations	accuracy	
layer		parameters		sample (%)	class (%)
vanilla	(48,96,192,256)	1,508,856	12	86.9	84.5
vanilla	(96,192,384,512)	5,951,432	12	87.8	85.8
vanilla	(96,192,384,512)	5,951,432	20	87.7	85.1
RCC-I	(48,96,192,256)	772,344	12	87.5	85.5
RCC-I	(96,192,384,512)	3,046,856	12	89.0	86.6
RCC-I	(96,192,384,512)	3,046,856	20	88.9	86.5
RCC-II	(48,96,192,256)	821,064	12	88.6	86.5
RCC-II	(96,192,384,512)	3,241,064	12	90.1	88.0
RCC-II	(96,192,384,512)	3,241,064	20	90.6	88.2

Three types of network architecture are investigated. We implement and train VoxNet following [13], and change it slightly to suit the normal vector input. As VoxNet uses 12-rotation input, we use the same input for the other two networks to enable the comparison between them. As can be seen, the normal vector input outperforms the voxel input consistently. Comparing the three networks, we find that the more sophisticated the network architecture is, the larger the performance difference between voxel input and normal vector input will be. It shows that sophisticated network has a strong capability to exploit the extra information provided by normal vectors.

We further study the per-class gain in classification accuracy when using RCC-II, and the results are shown in Fig. 6. About half of the classes have positive gains when changing the input from voxels to normal vectors, and classes with fewer training samples tend to have larger gain. The two classes *bowl* and *wardrobe* get a 20% increase in classification accuracy, while they have the fewest and the third fewest training samples. When using the voxel input they are apt to be misclassified as *flowerpot* and *dresser* respectively. The normal vector input makes these categories with similar appearance more distinguishable.

Evaluation of network configurations We study a variety of network configurations with the normal vector input, and the results are summarized in Table 2. For the numbers of channels n_1 to n_4, we investigate two combinations: $N_1 = (48, 96, 192, 256)$ and $N_2 = (96, 192, 384, 512)$. A steady improvement in the classification accuracy can be seen from N_1 to N_2, especially for RCC conv

Table 3
Classification accuracy on ModelNet40 with the combination of normal vector input and voxel input. Hard parameter sharing performs worse than no sharing, while CSNs do have the effect of improving the classification accuracy.

method	12 rotations		20 rotations	
	sample (%)	class (%)	sample (%)	class (%)
share(3)	89.6	87.5	89.8	87.4
share(3,4)	90.0	87.9	89.9	87.6
share(4)	90.2	88.4	90.3	88.0
no sharing	90.7	88.3	91.3	88.6
CSNs(channel)	91.0	88.5	91.7	88.8
CSNs(layer)	91.0	88.8	91.9	88.8

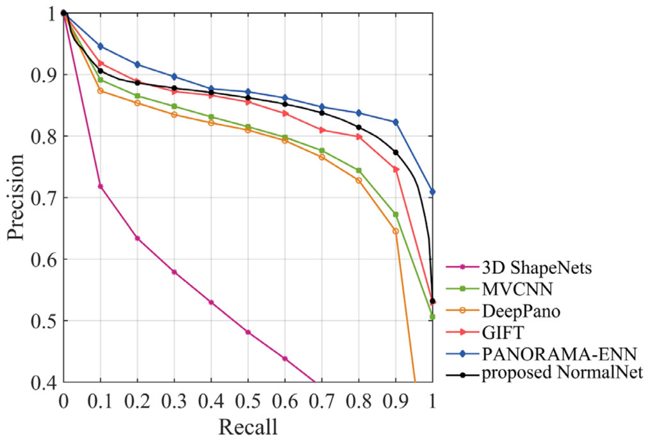

Fig. 7. Precision-recall curves of different methods on ModelNet40.

layers. It shows that an increase in the numbers of channels improves the performance. However, the use of more channels than N_2 does not yield a significant effect. Combination N_2 achieves an accuracy of about 97% on the training set, which implies the network fully exploits the information in the training data, and more channels seem to just cause overfitting.

Three types of conv layer structure are compared, and the RCC module shows a clear superiority. The numbers of trainable parameters in RCC-I and RCC-II are 51% and 54% of that of the vanilla conv layer respectively. RCC-II realizes one more conv layer than the other two structures with a small price, and gets a significant boost in performance.

The 20-rotation input does not show an advantage over the 12-rotation input with vanilla and RCC-I conv layers, but produces some improvement with RCC-II. Once again, we find the need to handle complex data with complex network.

Network fusion We use the RCC-II module to realize the conv layers in both N-net and V-net. The two networks are pre-trained separately, which can reduce the training time of the fused networks by about 50%. Hard parameter sharing can cover one or more layers from conv2, conv3, and fc4, and we test all combinations except share(2,4) (i.e. sharing conv2 and fc4) under the assumption that the shared layers should be consecutive. We also test the case of no parameter sharing, which is used as a benchmark. For the evaluation of CSNs, we test both layer-wise and channel-wise cases. The combination coefficients α_S and α_D (see [36] for the definition) are initialized to 0.9 and 0.1 respectively.

The results of different network fusion strategies and different inputs are summarized in Table 3. For hard parameter sharing, we have share(3) < share(3,4) < share(4) < no sharing in terms of the classification accuracy. This is true for both inputs. If the shared layers include conv2, the accuracy is even lower than that of share(3), and is not shown in the table. Since sharing any layer generates worse results than no sharing, we conclude that hard parameter sharing has no help for this task. On the other hand, CSNs improve the classification accuracy noticeably. It is somewhat confusing that layer-wise CSNs perform slightly better than the channel-wise case. Theoretically, this should not happen since the former is only a special case of the latter. It seems that channel-wise CSNs have too many combination coefficients and tend to fall into a local optimal solution.

We finally obtain the best classification accuracy in this work, which is generated by layer-wise CSNs, combining N-net and V-net, and with 20-rotation input. Note that the sample accuracy increases by 1.3%, whereas the class accuracy increases by only 0.6%, which suggests that the improvement mainly comes from classes with more samples.

Performance comparison We compare our method with state-of-the-art methods in Table 4. Overall, the performance of volumetric CNNs is worse than multiview CNNs and point cloud networks, except VRN-ensemble [5] which is a combination of 6 models. The proposed NormalNet performs inferior to VRN-ensemble, while comparable to other volumetric CNNs. To the network size, NormalNet has less than 6.5 M parameters (about 3.2 M for N-net and V-net each, and 16 for cross-stitching), and has 6 conv layers (2 for each RCC-II module) and an fc layer. The training of CSNs

Table 4
Classification accuracy of different methods on ModelNet40 and ModelNet10 (abbreviated as MN40 and MN10 respectively). The accuracy is averaged over classes. RT = randomized kd-trees, TA = translation augmentation, SA = anisotropic scaling augmentation.

Method	Input	Pre-train	Size	Augmentation	Accuracy	
					MN40 (%)	MN10 (%)
PointNet [38]	volumetric	–	80 M	–	–	77.6
3DShapeNets [4]	volumetric	MN40	38 M	12 rot.	77.3	83.5
VoxNet [13]	volumetric	–	0.92 M	12 rot.	83	92
ORION [21]	volumetric	–	4M	12 rot.	–	93.9
AniProbing [8]	volumetric	–	–	20 ori-pooling	85.6	–
Subvol. Sup. [8]	volumetric	–	16 M	20 ori-pooling	86.0	–
LightNet [25]	volumetric	MN10	0.3 M	12 rot.	88.9	93.9
VRN-ensemble [5]	volumetric	MN40	90 M	12&24 rot.	95.5	97.1
MVCNN [8]	multiview	ImageNet	VGG-M based	80 views	90.1	–
PANO-ENN [9]	multiview	–	–	–	95.6	96.9
RotationNet [11]	multiview	ImageNet	VGG-M based	12 views	**97.4**	**98.5**
FusionNet [6]	vol.+mul.	ImageNet	118 M	60 rot.	90.8	93.1
NormalNet (ours)	norm.+vol.	–	6.5 M	20 rot.	88.8	93.1
Kd-network [27]	points	–	–	RT+TA+SA	88.5	93.5
PointCNN [30]	points	–	0.45 M	random sample	91.7	–
SO-Net [31]	pt.+norm.	MN40	–	noise+scaling	90.8	95.5

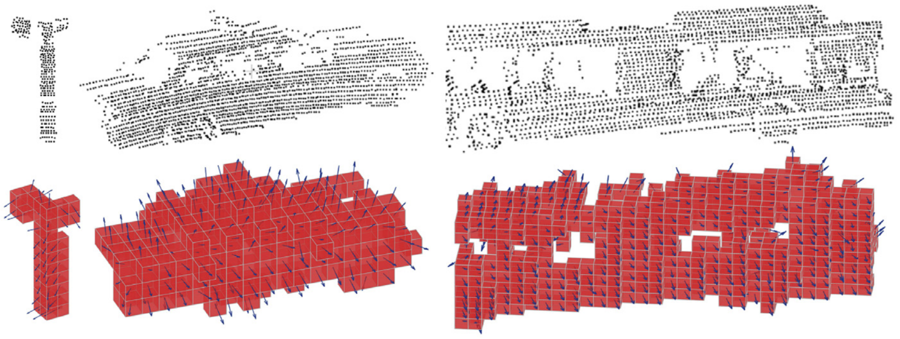

Fig. 8. Examples of labeled point clouds in Sydney Urban Objects Dataset (top) and the corresponding normal vectors (bottom). From left to right: traffic light, car, and bus.

Table 5
Retrieval performance of different methods on ModelNet40 measured in terms of mAP.

method	mAP (%)
3D ShapeNets [4]	49.2
MVCNN [10]	79.5
DeepPano [26]	76.8
GIFT [39]	81.9
NormalNet (ours)	84.4
PANORAMA-ENN [9]	**86.3**

Table 6
Classification performance of different methods on Sydney Urban Objects Dataset.

method	average F_1
UFL+SVM [40]	0.67
GFH+SVM [41]	0.71
multi-resolution VoxNet [13]	0.73
NormalNet (ours)	0.74
ORION Fusion [21]	0.78
LightNet [25]	**0.80**

takes about 9 h on Titan X, and roughly the same amount of time is needed to pre-train each net. The classification of each test sample takes about 5.6 ms.

4.2. CAD shape retrieval

Dataset and feature generation The retrieval performance of NormalNet is also evaluated on ModelNet40. The features are generated from the layer-wise CSNs with 20-rotation input, which achieves the best classification accuracy in Table 3. The activations of the two fc4 layers are concatenated to form a vector, and the vectors from all rotations of the same object are averaged to generate the original descriptor $\phi \in \mathbb{R}^{1024}$. A projection matrix W is trained to reduce the dimensionality to $p = 64$. For the evaluation, we treat each test sample as a query model and all of the samples in the test set as the target retrieval database. Note that the first retrieved model is always the query model itself.

Performance and comparison The retrieval performance is measured via mean average precision (mAP) and precision-recall (P-R) curve. The comparison between our method and state-of-the-art methods in terms of mAP is shown in Table 5, and the P-R curves of the methods are shown in Fig. 7. The curves of the compared methods are duplicated from the literature in which the resolution of recall is relatively low. The performance of our method is worse than PANORAMA-ENN [9], while better than other compared methods.

Effect of projection We test different values of p from 32 to 192 with interval 32, and we find that $p = 64$ achieves the highest mAP. However, the change of mAP for different values of p is less than 1%. It is worth noting that without the projection, the mAP is only 66.9%. The projection improves the mAP by about 17% and reduces the dimensionality of the descriptors to 1/16. We also calculate the ratio between the average intra-class distance and the average inter-class distance before and after projection. For the training set, these two ratios are 0.551 and 0.350, and for the test set, 0.573 and 0.437, respectively.

4.3. Point cloud object classification

Dataset and network settings The experiment is conducted on the Sydney Urban Objects Dataset,[2] which contains labeled Velodyne scans of 631 objects of 26 categories. The sizes of the objects range from one meter to dozens of meters. In the voxelization process, there is a tradeoff between retaining the overall shape and capturing sufficient spatial details. In [13], two voxel sizes (0.1 m and 0.2 m) are used to construct a multi-resolution VoxNet. We test several values and empirically select a voxel size of 0.3 m. We cut a $(9\ \mathrm{m})^3$ cube from the point cloud that contains the maximum number of points, and voxelize it to 30^3 occupancy grid (see Fig. 8 for examples). We use N-net with RCC-II conv layers for the classification, and set the numbers of channels n_1 to n_4 to $N_3 = (18, 36, 72, 96)$. Following [13], we use 18 rotations around the z-axis for data augmentation. Dropout is not applied.

Performance and comparison We follow the protocol employed by the dataset authors, which measures the performance using the average F_1 score, weighted by class support, for a subset of 588 samples in 14 classes over 4 training/test splits. We compare the performance of different methods in Table 6. Our method achieves an average F_1 of 0.738, marginally better than multi-resolution VoxNet, better than SVM-based methods [40,41], while lower than ORION [21] and LightNet [25]. We also tested voxel input, which achieves a slightly lower average F_1, i.e., 0.729.

5. Conclusion

We present NormalNet, a voxel-based CNN which exploits 3D representations for object recognition tasks. We use normal vectors as input, and boost the performance of the network from the aspects of network architecture, data augmentation, and network fusion. Our method achieves competitive performance in a series of experiments. The techniques used in this work can easily be extended to other volumetric CNNs. We use two reflections in each RCC, and the optimal number of reflections in an RCC module is an open question.

[2] http://www.acfr.usyd.edu.au/papers/SydneyUrbanObjectsDataset.shtml.

Acknowledgment

This work was supported in part by the National Key R&D Program of China under Grant 2016YFC1401001, and in part by the National Natural Science Foundation of China under Grant 61371144 and Grant U1605254. The authors acknowledge NVIDIA for providing a TITAN X GPU for the experiments involved in the research.

References

[1] A.X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al., ShapeNet: an information-rich 3D model repository, arXiv: 1512.03012 (2015).
[2] S. Choi, Q.-Y. Zhou, S. Miller, V. Koltun, A large dataset of object scans, arXiv:1602.02481 (2016).
[3] S. Song, S.P. Lichtenberg, J. Xiao, SUN RGB-D: a RGB-D scene understanding benchmark suite, in: Proceedings of the CVPR, 2015, pp. 567–576.
[4] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3D ShapeNets: a deep representation for volumetric shapes, in: Proceedings of the CVPR, 2015, pp. 1912–1920.
[5] A. Brock, T. Lim, J.M. Ritchie, N. Weston, Generative and discriminative voxel modeling with convolutional neural networks, arXiv: 1608.04236 (2016).
[6] V. Hegde, R. Zadeh, FusionNet: 3D object classification using multiple data representations, arXiv: 1607.05695 (2016).
[7] E. Johns, S. Leutenegger, A.J. Davison, Pairwise decomposition of image sequences for active multi-view recognition, in: Proceedings of the CVPR, 2016, pp. 3813–3822.
[8] C.R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, L.J. Guibas, Volumetric and multi--viekw CNNs for object classification on 3D data, in: Proceedings of the CVPR, 2016, pp. 5648–5656.
[9] K. Sfikas, I. Pratikakis, T. Theoharis, Ensemble of PANORAMA-based convolutional neural networks for 3D model classification and retrieval, Comput. Graph. 71 (2017) 208–218.
[10] H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller, Multi-view convolutional neural networks for 3D shape recognition, in: Proceedings of the ICCV, 2015, pp. 945–953.
[11] A. Kanezaki, Y. Matsushita, Y. Nishida, RotationNet: joint object categorization and pose estimation using multiviews from unsupervised viewpoints, arXiv:1603.06208 (2016).
[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: a large-scale hierarchical image database, in: Proceedings of the CVPR, 2009, pp. 248–255.
[13] D. Maturana, S. Scherer, VoxNet: a 3D convolutional neural network for real-time object recognition, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 922–928.
[14] G. Riegler, A.O. Ulusoys, A. Geiger, OctNet: Learning deep 3D representations at high resolutions, arXiv:1611.05009 (2016).
[15] J. Knopp, M. Prasad, G. Willems, R. Timofte, L. Van Gool, Hough transform and 3D SURF for robust three dimensional classification, in: Proceedings of the ECCV, 2010, pp. 589–602.
[16] I. Kokkinos, M.M. Bronstein, R. Litman, A.M. Bronstein, Intrinsic shape context descriptors for deformable shapes, in: Proceedings of the CVPR, 2012, pp. 159–166.
[17] R.B. Rusu, N. Blodow, M. Beetz, Fast point feature histograms (FPFH) for 3D registration, in: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), 2009, pp. 3212–3217.
[18] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, N.M. Kwok, A comprehensive performance evaluation of 3D local feature descriptors, Int. J. Comput. Vis. 116 (1) (2016) 66–89.
[19] S. Ji, W. Xu, M. Yang, K. Yu, 3D convolutional neural networks for human action recognition, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2013) 221–231.
[20] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei, Large-scale video classification with convolutional neural networks, in: Proceedings of the CVPR, 2014, pp. 1725–1732.
[21] N. Sedaghat, M. Zolfaghari, T. Brox, Orientation-boosted voxel nets for 3D object recognition, arXiv:1604.03351 (2016).
[22] J. Huang, S. You, Point cloud labeling using 3D convolutional neural network, in: Proceedings of the ICPR, 2016, pp. 2670–2675.
[23] J. Wu, C. Zhang, T. Xue, B. Freeman, J. Tenenbaum, Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling, in: Proceedings of the NIPS, 2016, pp. 82–90.
[24] A. Sharma, O. Grau, M. Fritz, VConv-DAE: Deep volumetric shape learning without object labels, in: Proceedings of the ECCV Workshops, 2016, pp. 236–250.
[25] S. Zhi, Y. Liu, X. Li, Y. Guo, Toward real-time 3D object recognition: a lightweight volumetric CNN framework using multitask learning, Comput. Graph. 71 (2017) 199–207.
[26] B. Shi, S. Bai, Z. Zhou, X. Bai, DeepPano: deep panoramic representation for 3-D shape recognition, IEEE Signal Process. Lett. 22 (12) (2015) 2339–2343.
[27] R. Klokov, V. Lempitsky, Escape from cells: deep Kd-networks for the recognition of 3D point cloud models, arXiv:1704.01222 (2017).
[28] C.R. Qi, H. Su, K. Mo, L.J. Guibas, PointNet: deep learning on point sets for 3D classification and segmentation, arXiv:1612.00593 (2016).
[29] C.R. Qi, L. Yi, H. Su, L.J. Guibas, Pointnet++: deep hierarchical feature learning on point sets in a metric space, in: Proceedings of the Advances in Neural Information Processing Systems, 2017, pp. 5105–5114.
[30] Y. Li, R. Bu, M. Sun, B. Chen, PointCNN, arXiv:1801.07791 (2018a).
[31] J. Li, B.M. Chen, G.H. Lee, SO-Net: self-organizing network for point cloud analysis, arXiv:1803.04249 (2018b).
[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the CVPR, 2015, pp. 1–9.
[33] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: Proceedings of the ICML, 2015, pp. 448–456.
[34] S. Ruder, An overview of multi-task learning in deep neural networks, arXiv:1706.05098 (2017).
[35] Y. Yang, T.M. Hospedales, Trace norm regularised deep multi-task learning, arXiv:1606.04038 (2016).
[36] I. Misra, A. Shrivastava, A. Gupta, M. Hebert, Cross-stitch networks for multi-task learning, in: Proceedings of the CVPR, 2016, pp. 3994–4003.
[37] K. Simonyan, O.M. Parkhi, A. Vedaldi, A. Zisserman, Fisher vector faces in the wild., in: Proceedings of British Machine Vision Conference (BMVC), 2013.
[38] A. Garcia-Garcia, F. Gomez-Donoso, J. Garcia-Rodriguez, S. Orts-Escolano, M. Cazorla, J. Azorin-Lopez, PointNet: a 3D convolutional neural network for real-time object class recognition, in: Proceedings of International Joint Conference on Neural Networks (IJCNN), 2016, pp. 1578–1584.
[39] S. Bai, X. Bai, Z. Zhou, Z. Zhang, L. Jan Latecki, GIFT: a real-time and scalable 3D shape search engine, in: Proceedings of the CVPR, 2016, pp. 5023–5032.
[40] M. De Deuge, A. Quadros, C. Hung, B. Douillard, Unsupervised feature learning for classification of outdoor 3D scans, in: Proceedings of the Australasian Conference on Robitics and Automation (ACRA), 2, 2013.
[41] T. Chen, B. Dai, D. Liu, J. Song, Performance of global descriptors for velodyne-based urban object recognition, in: Proceedings of the IEEE Intelligent Vehicles Symposium, 2014, pp. 667–673.

Cheng Wang received the Ph.D. degree in information and communication engineering from the National University of Defense Technology,Changsha, China, in 2002. He is a Professor with Fujian Key Laboratory of Sensing and Computing for Smart Cities and an Associate Dean of the School of Information Science and Technology, Xiamen University, Xiamen, China. His current research interests include remote sensing image processing, mobile LiDAR data analysis, and multisensor fusion. Dr. Wang has coauthored about 150 papers published in refereed journals. He is the Chair of the ISPRS Working Group I/6 on Multi-sensor Integration and Fusion (2016–2020). He is a Council Member of the Chinese Society of Image and Graphics.

Ming Cheng received the Ph.D. degree in biomedical engineering from Tsinghua University, Beijing, China, in 2004. He is currently a Professor with both Fujian Key Laboratory of Sensing and Computing for Smart Cities and Xiamen Key Laboratory of Geospatial Sensing and Computing, School of Information Science and Engineering, Xiamen University, Xiamen, China. His research interests include remote sensing image processing, point cloud processing, computer vision, and machine learning.

Ferdous A Sohel received Ph.D. degree from Monash University, Australia in 2009. He is currently a Senior Lecturer in Information Technology at Murdoch University, Australia. Prior to joining Murdoch University, he was a Research Assistant Professor/Research Fellow at the School of Computer Science and Software Engineering, The University of Western Australia from January 2008to mid-2015. His research interests include computer vision, image processing, pattern recognition, multimodal biometrics, scene understanding, robotics, and videocoding. He has published more than 80 scientific articles.He is a recipient of the prestigious Discovery EarlyCareer Research Award (DECRA) funded by the Australian Research Council. He is also a recipient of the Early Career Investigators award(UWA) and the best Ph.D. thesis medal form Monash University. He is a Member of Australian Computer Society and a Senior Member of the IEEE.

Mohammed Bennamoun received the M.Sc. degree in control theory from Queens University, Kingston, Canada, and the Ph.D. degree in computer vision from Queens/QUT in Brisbane, Australia. He is currently a Winthrop Professor at the University of Western Australia, Australia. He is the coauthor of the book Object Recognition: Fundamentals and Case Studies (Springer-Verlag, 2001). He has published more than 200 journal and conference publications. His areas of interest include control theory, robotics, obstacle avoidance, object recognition, artificial neural networks, signal/image processing,and computer vision.He served as a guest editor for a couple of special issues in International journals such as the International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI).He was selected to give conference tutorials at the Conference on Computer Vision and Pattern Recognition (CVPR 2016), European Conference on Computer Vision (ECCV 2002), the International Conference on Acoustics Speech and Signal Processing (ICASSP) in 2003 and Interspeech in 2014. He also contributed in the organization of many local and international conferences.

Jonathan Li received the Ph.D. degree in geomatics engineering from the University of Cape Town, Cape Town, South Africa, in 2000. He is currently a Professor with Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Information Science and Engineering, Xiamen University, Xiamen, China. He is also a Professor and the Head of the WatMos Lab, Faculty of Environment, University of Waterloo, Waterloo,ON, Canada. His current research interests include information extraction from LiDAR point clouds and from earth observation images. Dr. Li has coauthored more than 300 publications, over 130 of which were published in refereed journals. He is the Chair of the ISPRS Working Group I/2 onLiDAR-, Air- and Spaceborne Optical Sensing (2016–2020), the Chair of the ICA Commission on Sensor-driven Mapping (2015–2019), and an Associate Editor of IEEE Transactions on Intelligent Transportation Systems (IEEE-TITS) and IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing(IEEE-JSTARS).