SparseVoxNet: 3D Object Recognition with Sparsely Aggregation of 3D Dense Blocks

Ahmad Karambakhsh, Bin Sheng, *Member, IEEE*, Ping Li, *Member, IEEE*, Huating Li, Jinman Kim, *Member, IEEE*, Younhyun Jung, and C. L. Philip Chen, *Fellow, IEEE*

***Abstract*—Automatic recognition of 3D objects in a 3D model by convolutional neural network (CNN) methods has been successfully applied to various tasks, e.g., robotics and augmented reality. 3D object recognition is mainly performed by analyzing the object using multi-view images, depth images, graphs, or volumetric data. In some cases, using volumetric data provides the most promising results. However, existing recognition techniques on volumetric data have many drawbacks, such as losing object details on converting points to voxels and the large size of the input volume data that leads to substantial 3D CNNs. Using point clouds could also provide very promising results; however, point-cloud-based methods typically need sparse data entry and time-consuming training stages. Thus, using volumetric could be a more efficient and flexible recognizer for our special case in the School of Medicine, Shanghai Jiao Tong University. In this paper, we propose a novel solution to 3D object recognition from volumetric data using a combination of three compact CNN models, low-cost SparseNet, and feature representation technique. We achieve an optimized network by estimating extra geometrical information comprising the surface normal and curvature into two separated neural networks. These two models provide supplementary information to each voxel data that consequently improve the results. The primary network model takes advantage of all the predicted features and uses these features in Random Forest for recognition purposes. Our method outperforms other methods in training speed in our experiments and provides an accurate result as good as the state-of-the-art.**

***Index Terms*—3D convolutional network, 3D recognition, volumetric representation, surface normal, SparseNet.**

Manuscript received August 26, 2019; revised June 21, 2021; accepted May 05, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61872241 and Grant 61572316, in part by the National Key Research and Development Program of China under Grant 2019YFB1703600, and in part by The Hong Kong Polytechnic University under Grant P0030419, Grant P0030929, and Grant P0035358.

Ahmad Karambakhsh and Bin Sheng are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (Email: karambakhsh@sjtu.edu.cn; shengbin@sjtu.edu.cn).

Ping Li is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong; and also with the School of Design, The Hong Kong Polytechnic University, Hong Kong (Email: p.li@polyu.edu.hk).

Huating Li is with the Shanghai Jiao Tong University Affiliated Sixth People's Hospital, Shanghai 200233, China (Email: huarting99@sjtu.edu.cn).

Jinman Kim is with the Biomedical and Multimedia Information Technology Research Group, School of Computer Science, The University of Sydney, Sydney, NSW 2006, Australia (Email: jinman.kim@sydney.edu.au).

Younhyun Jung is with the School of Computing, Gachon University, Seongnam 13120, South Korea (Email: younhyun.jung@gachon.ac.kr).

C. L. Philip Chen is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China; and also with the Navigation College, Dalian Maritime University, Dalian 116026, China (Email: philip.chen@ieee.org).

I. Introduction

With the rapid development of 3D reconstruction and modelling techniques, the 3D model's repositories have become very large. These repositories may include a variety of 3D models that requires categorization. However, this is an arduous task to do manually, and thus, having automatic 3D recognition methods is necessary. There exists a large number of methods on 3D object recognition based on the use of point-clouds [1], [2], [3], [4], volumetric information [5], [6], [7], [8], or even multi-view images [9], [10], [11], [12]. Among them, using volumetric data, similar to the pixels in a bitmap that naturally encode the spatial distribution of 3D shapes, has shown very promising results.

These days, volumetric CNN's have already gained success in shape classification and retrieval [5], [6], [13], [14]. Although some details of shape structures will be disappeared during the voxelization procedure, the features encoding as global information is already sufficient for some simple classification tasks. However, to be more accurate, it is desirable to preserve as much detailed information as possible [15], [16], [17], and this has motivated recent papers such as [18] to work on high-resolution volumetric data. In some other works, to decrease the computational cost and memory consumption, they use either more complex data formats such as octree [19] or carefully designed convolutional operation [20]. However, there is an inevitable loss of details due to the sub-sampling in feature extraction. In addition to this, features extracted by these networks are not dedicated to the intended application.

In our approach, contrary to previous methods that using unsupervised learned features, the combination of useful surface features is used to achieve high-performance results. We concentrated on building an efficient system for predicting a fine-tuned object category on different types of 3D model datasets. It ought to be mentioned that the features are not hand calculated by us but anticipated as a port of the inference pipeline by the network itself. We illustrate this by assessing voxel curvature and voxel normal as the most crucial mesh surface characteristic features that provide superior results compared to those not utilizing them.

In our network, the revolutionary technique of sparsely aggregated convolutional networks [21] is redeveloped in three-dimensions. It is preventing our 3D convolutional network from having too many parameters. Although having a deeper neural network can enable the model to learn more information, it mainly causes to the vanishing gradient problem. In SparseNet technique, the gradient decay is avoided by

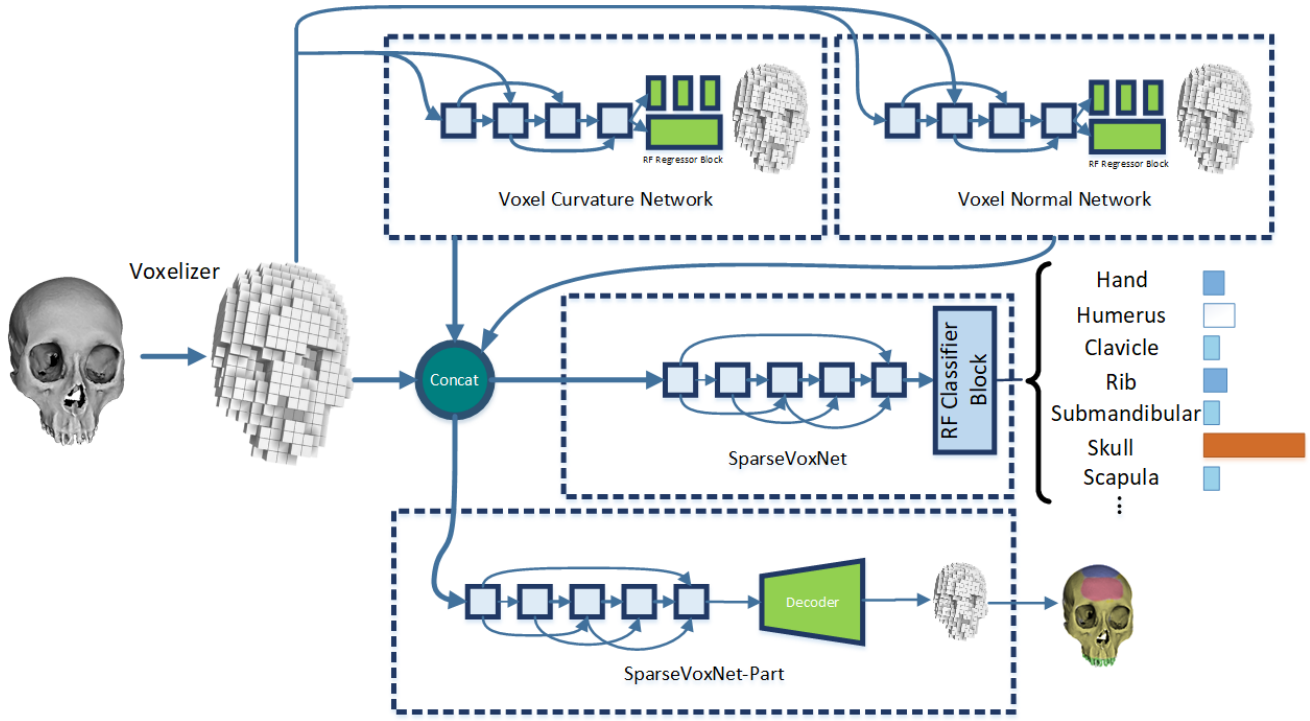


Fig. 1: The overall framework of our hybrid convolutional neural network for 3D object recognition purposed. In the first stage, we first converted an input mesh to a volumetric model (voxelized). Two network models have been developed to estimate the surface features like, voxel normal and voxel curvature, from the entry objects. Finally, the predicted results of the preliminary networks are merged into the volumetric data and feed into the third network for classification. All the network models have taken advantage of SparseNet topology for efficiency gains.

immediate access of the loss function to the essential layers. Besides, according to the main idea of using hand-engineered features, our proposed network decomposes into tree networks. The VoxNormNet (voxel normal network) and VoxCurvNet (voxel curvature network) are responsible for finding surface direction and the curvature of a bunch of input volumetric data. In our method, the main proposed model is responsible for the recognition of the input model, as it is shown in Fig. 1. The third network precisely employs features estimated from two preliminary parts and provides our predominant classification efficiency. To make the training and testing procedure even more efficient, we also took advantage of the Feature representation technique [22] in our three network models. In this method, features have to be extracted in sparse layers, and for the training of the feature, machine learning techniques, in our case, Random Forest, have been used to interpret the output even more accurately.

Our key idea in this work is to effectively employ two extra networks to preprocess the input data and estimate two useful hand-crafted features that significantly improve the third network's processing time, by providing more descriptive shape information. As some popular methods such as [23], [24] that employing surface normal in the two-dimensional learning process, our suggested method uses normal and curvature networks. However, we are providing these features on 3D volumetric data, which significantly improve our result on our 3D convolutional network. These provided features have more information to be identified in a recognition network. Besides, dividing a part of feature extraction to the other networks, assist the feature extraction method and leads to faster training. Consequently, the two preliminarily models can be trained separately, which makes our main suggested model even more efficient. To demonstrate the performance of our method, we also attempted for 3D model segmentation with an almost similar structure as the proposed recognizer method. To this purpose, we suggest an accurately inherited network from our recognizer named SparseVoxNet-Part. This network is designed to learn segmentation labels from similar recognizer data entries effectively. Without reducing the volume resolution, this model decreases the loss of detailed structural information without reducing the volume resolution generated by sub-sampling operations in CNNs, which is used in sparse 3D blocks and label decoder stage. Our work makes the following four main technical contributions:

- The SparseVoxNet is proposed, which is a novel 3D convolutional neural network with sparse connectivity between its network layers. The suggested structure provides higher efficiency and keeps our neural network away from vanishing gradient issues.
- Two other models are proposed to preprocess the 3D models for estimating important local features such as surface normal and curvature. Merging these features with voxel data, significantly improved the accuracy and efficiency of our suggested 3D recognition method.
- Using Feature Representation techniques to improve the classification and regression models. It means, CNN

model has to be trained once and then part of the model will be used to extract features and will be used in Random Forest for another training.
- We also expand the suggested recognition network to the 3D mesh part annotation network by replacing the last classification part of the model with a decoder stage to provide the segmentation results.

II. Related Work

A large number of researches in the computer vision and graphics fields community has been devoted to constructing a way of recognizing 3D objects. Different representations are used to describe these 3D models, there are some works on shape descriptors for the voxel and projected view representations, in some other researches. Here we describe some of the publications which have some improvement in this field.

Shape descriptors There are various methods related to the video analysis and image understanding challenges that utilize different descriptors to provide more accurate classification [25]. For instance, Czajewski and Kołomyjec [26] published a stunning paper in 3D object recognition based on RGB-D images. They proposed a method employed VFH (Viewpoint feature Histogram) and CRH (camera roll histogram) as their descriptor, and ICP (Iterative closest point) was the main matcher afterward. As they mentioned, their recognition performance was even better than the CNN-RNN method of Socher et al. [27], which introduced a model that combines convolutional and recursive neural networks (RNN) for learning features and classifying RGB-D images. Gomes et al. [28] proposed the usage of a moving fovea approach to down-sample 3D data and reduce the processing of the object retrieval system from point clouds. They mentioned their object recognizer could work $7\times$ faster than a non-foveated approach. The main idea was that the point density is higher close to the fovea and that this density decreases according to the distance from the fovea, which means they can be reducing the number of points and processing time at the same time.

Convolutional descriptor Convolutional neural networks (CNN) have achieved the best performance in many computer vision tasks, including object recognition such as large-scale classification [29] and action recognition [30]. By jointly encoding convoluted information in the learning process, 2D convolutional networks have achieved state-of-the-art performance in object detection and classification. Other researches use 3D CNN structures to perform recognition and detection tasks in a video by tuning the networks using video frames [31]. Gkioxari and Malik [32] proposed an action detection method that trained to detect bounding boxes of actions frame-by-frame from a video. VoxNet [5] by Maturana and Scherer focused on LIDAR and RGB-D cameras to increase the robot conception in a real environment. Their proposed method, integrating a volumetric occupancy grid representation with a supervised 3D Convolutional Neural network. We can mention VoxNet as a foundation of many other proposed methods now. The other similar work is Wu et al. [6], which focused on extracting a volumetric representation of a 3D model from a dataset of 2.5D range data. This method achieved an exciting result in depth sensor such as Kinect on its publication time.

At the same time, Su et al. [9] suggested to render 12-views for 3D meshes and classify the rendered images. To do the image classification, they have used VGG [33] which is already trained on ImageNet data [34]. The MVCNN-MultiRes [35] improved MVCNN by using rendered images from a variety of resolutions. FusionNet [36] tackled the 3D object recognition of ModelNet dataset using two data representation: Volumetric representation and Pixel representation. They combine two different voxel CNNs with a single multi-view network, which could achieve accuracy 92.11 on modelnet10 and 90.8 accuracy in modelnet40, which was the highest recognition accuracy in 2016. Supervised training from labeled mesh datasets is achieved on mesh segmentation and part labeling by Yi et al. [37]. However, such methods rely on annotated databases of segmented meshes, which is an extremely labor-intensive process. Capturing the right scale part is very difficult with non-expert manual annotation. Besides, SyncSpecCNN [38] is investigated on a spectral CNN learning on a graph of triangulated vertices by Yi et al.

Additionally, VoxelNet [39] by Zhou and Tuzel had an investigation in 2017 using a CNN method and voxelizing data to achieve an accurate 3D recognizer. The results were achieved by not only the classification result but also to localize the objects in The Kitti dataset. Zhi et al. [40] introduced a real-time method of predicting class label and orientation information simultaneously without additional annotation. They designed a shallow network competing for 3D object recognition accuracy with some of the state-of-the-art techniques in training parameters. Furthermore, there is another type of system that concentrated on different connectivity of the network blocks [41], [42], which can address some of the challenges efficiently. In the mean time, submanifold sparse convolutional networks [18] is also introduced to process the sparse data efficiently.

Liang et al. [43] concentrated on 3D object recognition and pose estimation of multiple projected view of a 3D model. They have used two Deep Belief Networks to extract the image features, and they connect the last layer to match features and classify the input data. Besides, they apply a new deep brief network that combines the two traditional DBFs and estimates the related position similar to classification issue. They also presented the K-means clustering to overcome the shortcoming in object detection, which ends to the accurate result. PointNet [1] could not recognize local features by the metric space points, limiting its ability to extract fine-grained patterns, and It has difficulties in complex scenes. Qi et al. [2] introduced a hierarchical neural network that used PointNet recursively on a nested partitioning of a point cloud. By applying metric space distances, the system is capable of learning local features with increasing contextual scales; this method is called PointNet++. Liu et al. [44] proposed to use volumetric representation and unsupervised deep learning networks to extract the features of point cloud data directly. They also applied the Hough Forest method on the extracted features and achieve object detection and pose estimation simultaneously. They compared their results based on Tejani et al. [45] dataset of 2.5D data

and reached 0.741, which was almost acceptable. Ma et al. [46] studied a new type of volumetric CNN named binary volumetric CNN, which could decrease the model parameters and make the network much more efficient.

Wang et al. [47] depicted the EdgeConv layer in deep learning, capturing local geometric features of point clouds. The entire architecture followed the PointNet architecture except applying EdgeConv Blocks that has a considerable effect on the results. They performed 92.2% accuracy for classification of ModelNet40, which was higher than the state-of-the-art such as VoxNet, PointNet++, and KD-Net. Accordingly, Xie et al. [48] concentrated on using nonlinear distance metric between 3D shape descriptors for retrieval. They evaluated their data on McGill, SHREC'10, ShapeGoogle, and SHREC'14 datasets, which are different from datasets we required to use. Also, several 2D recognition methods have been studied in terms of investigation on recent research in recognition topics. For instance, if we concentrate on image recognition, RVM (representative vector machine) [49] is highlighted, focusing on character recognition PC-2DLSTM (principal component 2-D long short-term memory) [50]. And, metric-learning-based recognition [51] achieved the precise result using a deep neural network. In terms of face recognition and facial expression, SSP (superimposed sparse parameter) classifier [52] and AFERS (automatic facial expression recognition system) [53] have been proposed, and their classifiers are among the top methods.

Feature Representation The idea of using CNN features in different techniques of machine learning is not new. The activations, which are the output of CNN layers, can be interpreted as features. Trained CNN models for classification can be used as feature extractors by removing the output layer. There are a variety of techniques in supervised learning, such as SVM, SGD, Random Forest, and decision-tree. Among them, the Random Forest(RF) algorithm has been extensively applied for classification or regression tasks by building a large number of classification or regression trees (CART) and combining bootstrap and aggregation ideas [22]. When RF is used for regression problems, the output variables are fitted by using samples of the input variables. For each of the input variables, the data are divided into several points, and the Sum of Square Error (SSE) is calculated at each divided point for predicted and actual values. Then, the minimum SSE value is chosen for this node. In addition, the variable importance can be obtained by permuting all the values of the input variables and measuring their difference in prediction accuracy [54]. In the forest generation process, the number of decision trees N and the candidate split attribute value M are two parameters that notably influence the performance of the model. According to the law of large numbers, when the value of N increases, the generalization error converges, which can effectively avoid data over-fitting, but increasing this value after it reaches a certain threshold does not increase the accuracy of the model [55]. The idea of using feature extracted by CNN in Random Forest or other classifiers has shown a variety of benefits on different applications such as face recognition [56], software defect prediction [57], and even Genomic Science [58].

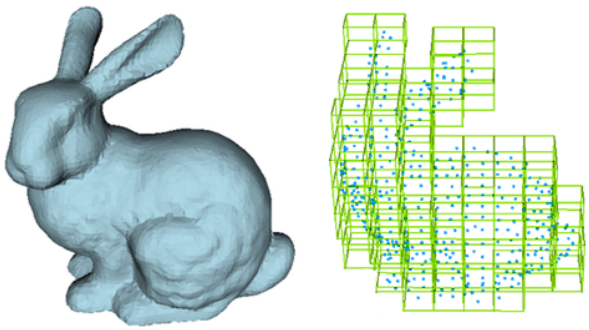


Fig. 2: Voxelizing a scanned 3D model can be much more efficient in the suggested voxelizer than interpolation or any other methods. It shows a point cloud generated from the scanner, which can directly be used in the voxel grid method. The left side shows a bunny 3D model, and the right side is the voxelized model generated directly from the 3D grid.

III. Approach Overview

Fig. 1 depicts an overview of the proposed novel volumetric object recognition and segmentation system. The proposed system consists of three main components in each task: one of the networks is provided to construct the 3D voxel normal information from the input volume. The second one is responsible for estimating the voxel curvature from the volumetric data. The third one utilizes these important surface features along with voxel information to precisely achieve the task of 3D model recognition. Moreover, after training the network, Random Forest regressor has been used to learn features more effectively. Another sparse network model uses the two preliminary network's output and voxel data to provide segmentation results.

A. Surface Feature Extraction

Using just volumetric data as the input is not sufficient to train and test the suggested recognition. In the proposed method, two preprocessing novel networks are responsible for estimating two vital surface feature voxel normal and curvature. These networks receive a window of volumetric data as their input and estimate their output layers' expected features. To improve it, the feature representation technique has been used. Having normal and curvature features as the entry for the classification model would result in more efficiency, which is explained in the result section.

B. Object Recognizer

After preprocessing the volumetric data using VoxNormNet and VoxCurvNet networks, the extracted data have to be concatenated with raw volume information to feed the third network. The SparseVoxNet as the third network is responsible for processing the merged features effectively and estimates the entry object category. The main structure of SparseVoxNet contains two stages: one is the deep sparse connected dense blocks and fully connected layers to train the input volume data once, and the second one is to use the part of the trained

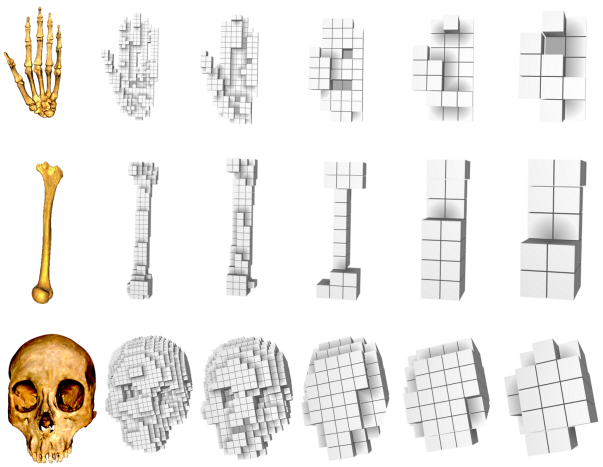


Fig. 3: Three different meshes of the human skeleton are shown that converted to volumetric data with different resolutions. It is clearly shown that the high-res voxel grid can keep the details more than low-res grids. It has to be mentioned that previous methods are designed to employ low-res volume to be capable of processing large amounts of data.

model for feature extraction and train the Random Forest classifier to reach a higher accuracy than using the network model individually.

C. Object Part Annotator

A similar structure is provided to reach the segmentation result on a 3D model dataset. The only and main difference between recognition and part segmentation networks is the last part of the structure, which carefully replaced the classification part with a 3D convolutional network to reconstruct the same size volumetric information as the input data.

IV. 3D Object Recognition

The main superiority of our recognizer is to take advantage of some hand-crafted features that have information from the surface points, such as normal which illustrates the surface point's direction, and curvature which is providing information of surface shape. The proposed method is efficient in train and test stage by having a modern technique in the connection between its layers. This section will outline all the steps from the recognition section, such as training on the input dataset and predicting the test entries.

A. Voxelization

Voxelization of a 3D model is the process of converting the geometric information of a mesh into a discrete domain grid. The method has to be chosen according to the type of 3D models. In this section, the technique of voxelizing ShapeNetCore-part dataset is explained, which contains unstructured point clouds similar to the 3D scanned objects. To process this point cloud data, we need to specify the grid size as the first step. The 3D positions of the selected 3D model have to be converted to the grid size domain, and consequently,

Algorithm 1 Voxelization

Require: Position data in 3 axis P_x, P_y, P_z , g_s
Ensure: Voxelized data V
procedure NORMALIZATION(I)
  find min_I;
  Decrease min_I from I;
  find max_I;
  $Out = I/max_I$;
  return Out;
end procedure
[N_x, N_y, N_z] = Normalize([P_x, P_y, P_z]);
$Ind_x = N_x \times g_s$;
$Ind_y = N_y \times g_s$;
$Ind_z = N_z \times g_s$;
V = Initialize a 3D Array(g_s,g_s,g_s);
Set $V[Ind_x, Ind_y, Ind_z]$ to the value of 1;

the founded grid-cells has to be triggered. Each grid position that is found should be set with the value of one, and the other would be zero. As an example of our voxelization, Fig. 2 illustrates a sample voxel data of the bunny 3D object and Fig. 3 represent three parts of the human body skeleton that directly achieved from the 3D grid of the object. The procedure of this approach is shown in Algorithm 1, where the P_x, P_y, P_z are the points from our point clouds, and g_s represents the grid size. In Algorithm 1, at the first step, position data should be normalized (between 0-1) and the multiplication of the g_s, by Ind_x, Ind_y, and Ind_z provides values that directly point to the index of each voxel grid. Thus, the V as the Voxelized output data contains a 3D matrix with a specific size and binary values. Each voxel represents a sample on a three-dimensional grid-cell. This procedure is developed using matrix calculation to be more optimized, fast, and able to process on tensors. The designed method is a simple procedure that is demonstrated in Algorithm 1.

B. Normal and Curvature Estimation

The surface normal is the essential properties of a 3D model. Normal vector estimation is also important in the calculation of curvature and surface reconstruction. There are two approaches to normal estimation: Firstly, the surface must be obtained by meshing techniques, and then the provided surface mesh is used to compute surface normal. Secondly, whenever the data of surface mesh is not reliable, normal data have to be achieved by approximation on point cloud directly. Our proposed approaches are reaching the point normal by using either of them. The first one would be beneficial when dataset containing surface mesh, and the other one is more likely to be used on a dataset containing unstructured point clouds.

Many different normal estimation methods are investigated in terms of estimating normal using point cloud directly. The point normal is determined according to [59], and they take this issue as a least square plane fitting estimation. Thus, this estimation can reduce to eigenvectors and eigenvalue of the covariance matrix, and the principal component analysis

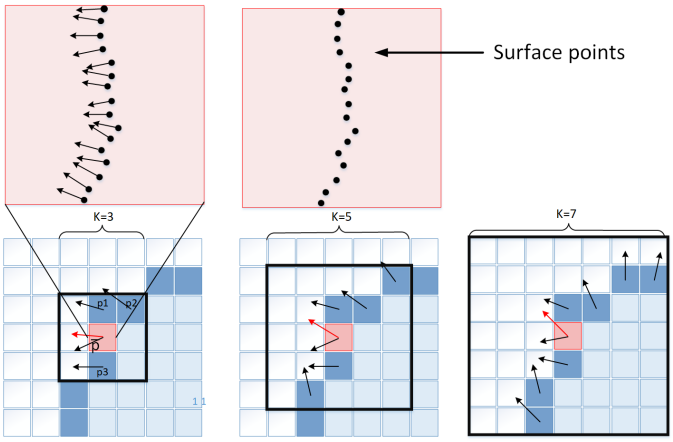


Fig. 4: Two-dimensional illustration of the averaging surface normal in a grid cell. From left to right, the K parameter changes to 3, 5, and 7. Decreasing the number of neighbours would provide voxel normals in more details. The red arrows show the average normal direction.

(PCA) would be an efficient method. In this technique, for each point p_i, we consider covariance matrix M via Eq. (1):

$$M = \frac{1}{K}\sum_{i=1}^{K}(p_i - \bar{p}).(p_i - \bar{p})^T, C.\bar{v}_j = \lambda_j.\bar{v}_j, j \in \{0,1,2\} \quad (1)$$

where, p_i is considered as a point on the surface, $\bar{p}$ is the centroid of the nearest neighbours, K represents the number of point neighbours, λ_j is j-th eigenvalue and $\bar{V}_j$ shows the j-th eigenvector. The issue of determining the sign of the normal is ambiguous, and there is no mathematical way for this problem. Still, in this issue which is considering point cloud to estimating surface normal, using PCA and extended Gaussian image (EGI) are well known. However, when the viewpoint is already known, no one has to follow those complex computation. Instead, the normal cloud orients consistently toward the viewpoint, and we have to satisfy Eq. (2) as:

$$\vec{n}_i.(V_p - p_i) > 0 \quad (2)$$

The other challenge is determining the K parameter or the radius parameter to finding the nearest neighbour of the selected point because a surface normal at a point should be estimated from its adjacent points. Defining K is constituting a limiting factor in the normal estimation, and it is the crucial parameter to reach the actual point's normal. Therefore, this parameter has to be selected based on the detail of the point cloud. Hence, the scale parameter is needed, and it has to be small enough to capture tiny details and be large enough to skip unnecessary data. We also provide another stage to average the normal data through the voxels, which assist in reaching more smooth normal data and ignoring the tiny normal changes. Fig. 4 shows the stage of averaging on 2D voxel normal data which is customized by K parameter that defines the number of the neighbor grid cell to compute the average. This stage can abstract the estimated data to comprehensive and trainable information. At the top side of this figure the surface pointclouds and normals are converted to a grid cells, to be ready for the stage of averaging. After preparing the normal information, the mean data of normal has to be calculated in each voxel grid that we call it voxel normal.

TABLE I: The network structure of SparseVoxNet. Multiple DB layers (Dense-Block layers) are responsible for extracting features in a very efficient way of sparse connectivity between them. The second stage of the network is the classification stage that receives the information from DB layers and estimates the input object class.

Layers	Input Shape	Output Shape	In
Input1	50×50×50×1	50×50×50×1	
Con3D1	50×50×50×1	6×6×6×16	[Input1]
DB1	6×6×6×16	6×6×6×16	[Con3D1]
Concat1			[Conv3D1], [DB1]
DB2	6×6×6×32	6×6×6×16	[Concat1]
Concat2			[DB1], [DB2]
DB3	6×6×6×32	6×6×6×16	[Concat2]
Concat3			[DB2], [DB3]
DB4	6×6×6×32	6×6×6×16	[Concat3]
Concat4			[DB1], [DB3], [DB4]
DB5	6×6×6×48	6×6×6×16	[Concat4]
Concat5			[DB4], [DB5]
BatchNorm1	6×6×6×32	6×6×6×32	[Concat5]
Activation1	6×6×6×32	6×6×6×32	[BatchNorm1]
Conv3D2	6×6×6×32	6×6×6×16	[Activation1]
Flatten1	6×6×6×16	3456	[Conv3D2]
Dense1	3456	256	[Flatten1]
Dense2	256	64	[FC1]
Dense3	64	10	[FC2]
Activation2	10	10	[FC3]

The extracted voxel normal is the target information of the VoxNormNet. The estimation by VoxNormNet can provide normal voxel information much faster than conventional methods to provide it from raw point clouds. This network is shown in Fig. 5. In the end, the result of the voxel-normal can be averaged, one more time, to provide more smooth and continuous normal information on the voxel structure. Generally, when the voxel-normal is calculated, they consider tiny details of the surface, and these details would affect the voxel-normal to be discontinuous and inaccessible from our network to train. But, with the averaging technique of the voxel-normals, we can prepare much more smooth and trainable voxel-normal, which is shown in Fig. 4 in two dimension. A Part of our implementation is done using PCL Library in C++ and Meshlab scripting. This part our coding provides volumetric data generation, surface-normal , and curvature. But another part which is responsible to generate our suggestion in voxel-normal and voxel-curvature is purely implemented in Python scripting.

C. Deep Neural Network Structure

In this work, we propose the SparseVoxNet, a 3D convolutional network that integrates a number of 3D convolutional layers through sparse aggregated connectivity [41]. The sparse connectivity enables our network to have deeper network layers without losing the value of the neurons (Vanishing Gradient). This structure not only provides an opportunity to have a deeper network but also having an efficient network and accurate results. The other paper inspiring our suggested

TABLE II: The dense block structure is the block that mentioned in Table I as DB. Sparse Connectivity of the dense blocks can improve the estimation accuracy with very few parameters rather than previous methods that handle the performance in very large networks.

layers	Input Shape	Output Shape	In
BatchNorm1	6×6×6×16		
Relu1	6×6×6×16		[BatchNorm1]
Conv3D1	6×6×6×16	6×6×6×64	[Relu1]
BatchNorm2	6×6×6×64		[Conv3D1]
Relu2	6×6×6×64		[BatchNorm2]
Conv3D2	6×6×6×64	6×6×6×16	[Relu2]
DropOut1	6×6×6×16		[Conv3D2]

method is DepthMapNet [60], which was followed from DenseNet [61] in its network's structure, estimates the dense-map of the input stereo images. The DenseMapNet follows the same layer connectivity as DenseNet, but, it is suggested to use a series of provided network layers called dense blocks, which could extract vital information from the input data. We redesign the dense block in three-dimensions, but rather than using dense connectivity, the sparse connectivity has been used, which shows much more efficiency in its provided results. Fig. 6(d) demonstrates the suggested 3D dense block in the proposed method. As we have already mentioned, there are three main networks in the suggested method:

- Network that estimates voxel normal (VoxNormNet)
- Network that estimate voxel curvature (VoxCurvNet)
- Network that employs the prepared volumetric inputs to classification purposes (SparseVoxNet)

Fig. 5 demonstrates our proposed advanced network structure to preprocess the volumetric input and extract voxel normal and curvature. The suggested network is a supervised method; the network requires to receive the ground truth normal voxel as the target data. The procedure of arranging these input data explains in Section IV.B, which is a calculation of normal and curvature and averaging them in a voxel grid structure. The third network, SparseVoxNet, is responsible for the classification of the objects, and it contains two main stages. The first stage is a sequence of the 3D dense blocks that are thoughtfully connected based on SparseNet [41]. The second part is having a series of convolutional layers to abstract more highlighted features and two fully connected layers to classify the information according to the target data. The detailed description of the main network is shown in Table I. Each Dense Block in Table I contains series of CNN layers which is shown in Table II. We implement our networks based on the Keras platform and use its element-wise product and convolution operation for our SparseVoxNet implementation.

Dividing the network structure to three separate networks has two advantages: Firstly, it prevents our model from being very large, and consequently, the requirements of memory and processing units will be reduced. Secondly, having fewer parameters in the recognizer's network can lead to a faster training procedure. Generally, the suggested method has demonstrated more efficiency by utilizing hand-crafted features as its entry. Fig. 6(c) shows the SparseVoxNet receiving

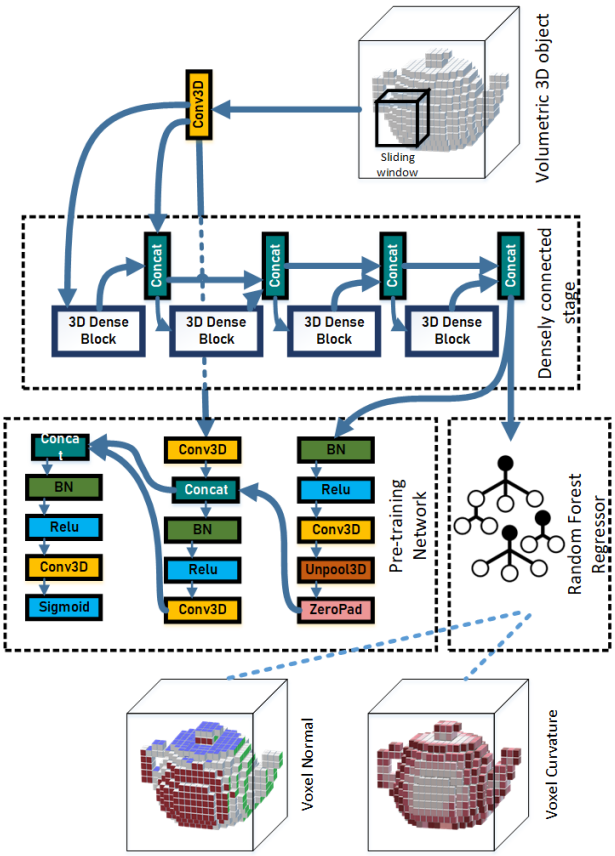


Fig. 5: Our network trains supervised and estimates voxel normal and voxel curvature with the same size as the input voxel. Densely connected layers assist the model in extracting significant related features in a compact size, and the decoder stage interprets the features and reconstruct the volumetric data containing the normal or curvature information. The predicted results of the preliminary networks are merged into the volumetric data and feed into the third network for classification. After training of each model, we train the Random Forest with features extracted from by the first part of the network.

its inputs from voxelizer, VoxNormNet, and VoxCurvNet. In this network, the Adam function is used as an optimizer. Almost all the activation functions are Relu (Rectified Linear Unit), except the last one, which is the SoftMax function.

D. Feature Representation

We consider training Random Forest with features provided by our sparse 3D block layers to provide even more accurate results. It has already been studied that using some of the machine learning techniques to interpret the features provided by CNN would provide more accurate results than using dense layers in a neural network. Athiwaratkun and Kang [22] demonstrated series of experiences that shows that Random Forest can outperform the others to interpret features provided by CNN. Therefore, at the first stage, the model has to be trained fully on the neural network model. After the training, the first stage of the trained model can be used to train random forest. We use our suggested CNN to extract features for

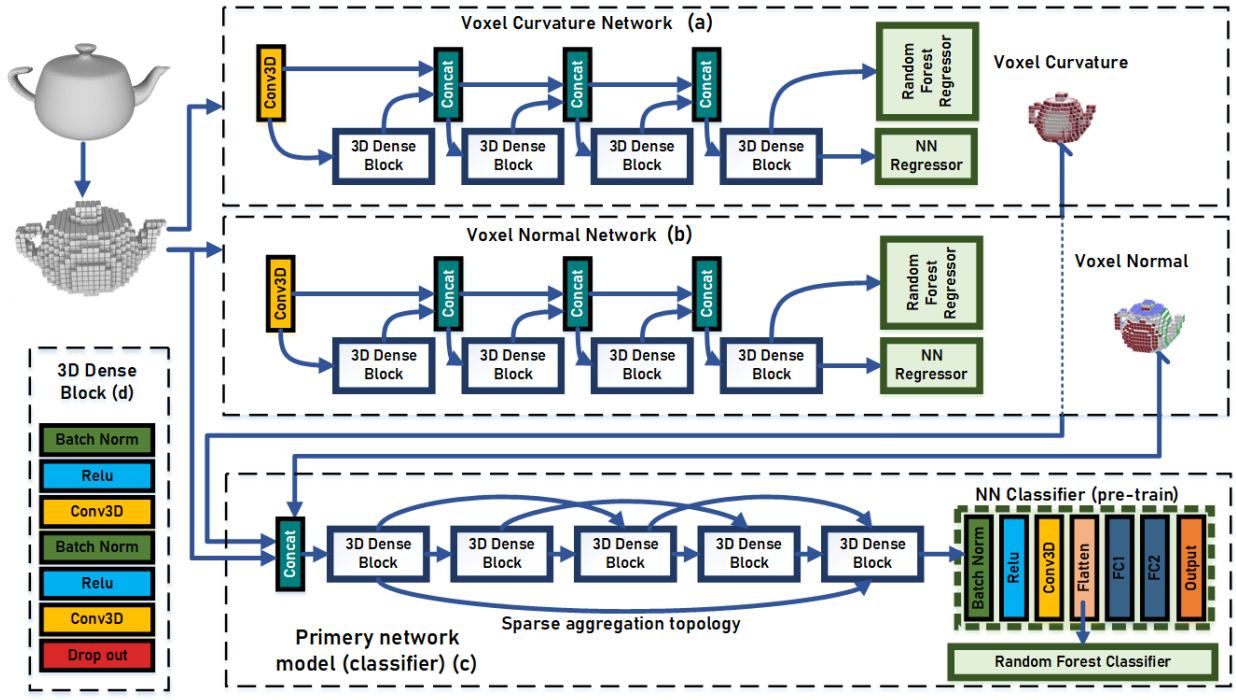


Fig. 6: The network structure of the proposed approach containing three different convolutional networks. Two preliminary networks are designed to preprocess the raw volumetric data to estimate the voxel normal and voxel curvature. The third network is provided to receive the information from voxelizer and preliminary networks to classify the input 3D object.

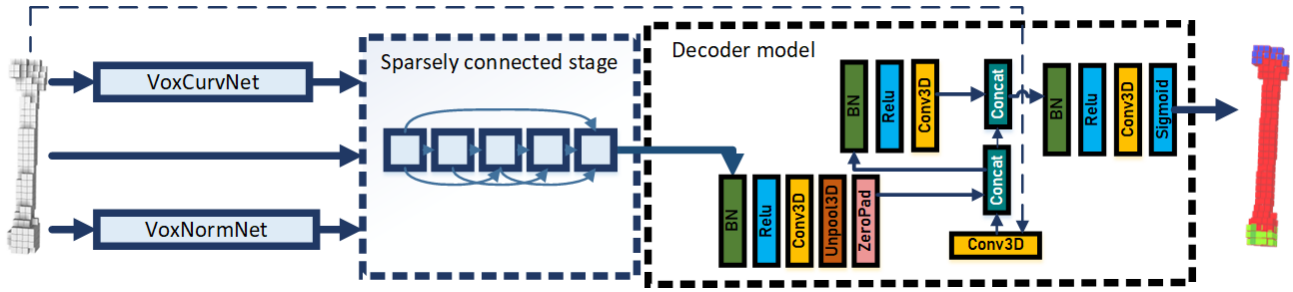


Fig. 7: The overall procedure of 3D object part annotation via the proposed approach. The network learns all the 3D volume labels through one training level. It means the output of this network would be the same voxel size data as the input data. VoxNormNet, VoxCurvNet, and sparsely connection stage are similar to Fig. 6. The contribution of estimated hand-crafted features and volumetric data shows undeniable progress in part segmentation challenge. Each model will be trained twice as we consider using the first train's features for the second train using Random Forest.

Random Forest that can interpret the results more accurately than fully connected layers in CNN models.

V. 3D Model Segmentation

To verify the suggested strategy, the method has been re-designed to fit on the part-segmentation challenge. The overall architecture of the part annotation approach is explained in Fig. 7, which contains two well-designed preprocessing networks, similar to the recognition approach, and a network that employs provided normal, curvature, and raw volumetric data to learn the segmentation. The suggested method is using almost the same procedure as the proposed recognizer. However, the network uses an optimized decoder stage to reconstruct the volumetric segmentation data instead of the classification stage. The main structure of the suggested method is illustrated in Fig. 7. Same as the proposed recognition method, the segmentation approach includes two extra networks to estimate voxel normal and voxel curvature, which then will be employed by the segmentation network. Fig. 6(a) and Fig. 6(b) illustrates the VoxNormNet and VoxCurvNet, which are already explained in the section of 3D recognition. These networks are just useful when the dataset has the ground truth normal and curvature information. The only difference between VoxNormNet and VoxCurvNet is in their output layers. The output layer in the VoxNormNet requires to have three channels representing the normal directions by three values of nX, nY, nZ. But the VoxCurvNet output requires one channel that belongs to curvature value. The structure of these networks is demonstrated in Fig. 5 in more details.

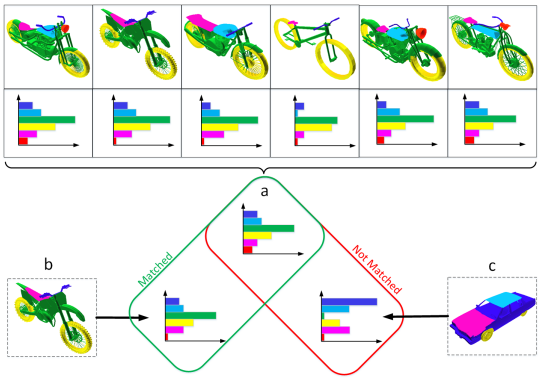


Fig. 8: To distinguish a wrong classification result of the deep neural network, the mesh part's histogram is used, which is shown in this figure. The mean of the ground truth data category should be compared to the new categorized data.

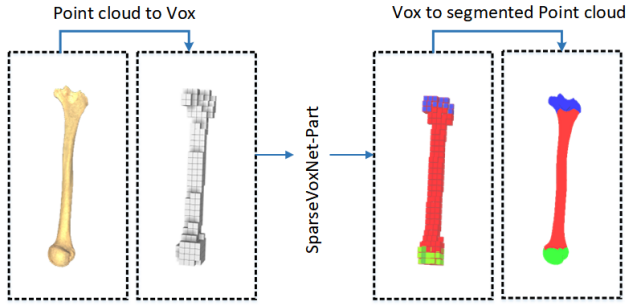


Fig. 9: The input point cloud is converted to volumetric data at the first step, and in the end, the segmented volume would switch to a point cloud.

The main section of the network is shown in Fig. 7. This network gathers all the preprocessed useful information and considers all of them to annotated the parts of the entry model. When the voxel normal and voxel curvature are prepared, they will be merged with the raw binary volume data. The merging process prepares a voxel with 50x50x50 size and five channels, which include the binary volume, voxel normal (three channels), and curvature(one channel). The third network that we call it SparseVoxNet-part, contains a series of dense block layers with sparse connections according to the SparseNet [41] method. But contrary to the recognizer, the second part of this network is not a classification stage, and it contains another decoder network similar to the VoxNormNet to rebuild a volume data with segmentation values.

The SparseVoxNet-part is prepared to demonstrate the performance of the suggested network to 3D segmentation purposes. Furthermore, the segmentation results can provide more information to assist the recognizer network; the result of segmentation in each class is almost unique, which makes them capable of being employed in the recognizer and verifying the final classification results. In this paper, the method prepares a histogram of the segmentation result without applying them to the automatic recognizer. The procedure contains segmenting the object according to the recognized class label and compare the histogram of the segmented object with the

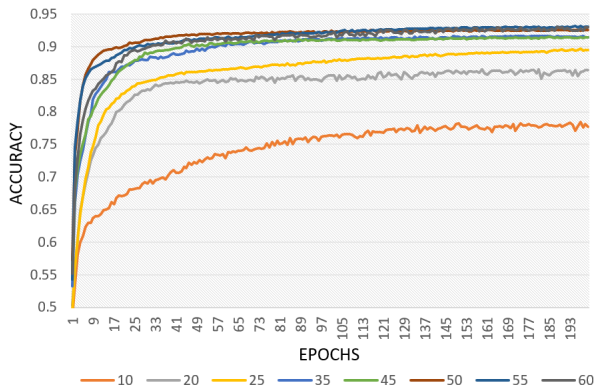


Fig. 10: Accuracy results by applying grid size from $10\times10\times10$ to $60\times60\times60$ in 200 epochs of training. It shows that the accuracy will increase by a larger grid size, but increasing it to more than 50 not only does not have a significant implication on the accuracy but also negatively affects the duration of the training process.

average histogram of that class, which is demonstrated in Fig. 8. This stage has not been used to improve our accuracy in comparison with other methods and only provides more trustability in detecting the wrong recognition. We provide this information to comfort the School of Medicine, Shanghai Jiao Tong University, enabling them to verify their classification results manually. Another function that is provided in our method is to converting the segmented result to point cloud. This function is almost the reverse procedure of Algorithm 1. Fig. 9 illustrates a point cloud that has to be converted in volume for the segmentation process and again converted to point cloud after segmentation for comparison purposes.

VI. Experimental Results

Dataset: In our suggested approach, two datasets are used. Firstly, the ModelNet10, a dataset prepared by Princeton University, was used as a test-bed for 3D recognition methods. The data contained 4899 objects across ten categories. The ModelNet10 dataset is already split to train and test objects. Secondly, the ShapeNetCore-part, which is one of the most popular datasets in terms of 3D object segmentation, was used that already provided normal and curvature information; this dataset was used for segmentation evaluation. The comparison of 18045 shapes across 16 categories has done. Both the datasets were used for recognition evaluation. Train-test split is considered according to [37].

Device: The lightweight structure of the suggested method provides us to train the network on a consumer PC; with Intel Core i7 4 GHz processor, NVIDIA Geforce GTX 1050Ti, 16 GB RAM, with 64-bit Windows 10.

A. Voxelization Configuration

To explore the parameters in the proposed method, we conduct a series of experiments on the analysis of the different size of the voxel grid. For this investigation, the ShapeNetCore-part dataset is used, which contains 16 different categories that

TABLE III: Classification accuracy in comparison with some of the state-of-the-art methods on ModelNet10. The results show our superior training speed, which makes it very easy to retrain and adapt to a new dataset.

Methods	Accuracy (%)	Size (M)	Training time	Inference time	Device
DeepPano [62]	85.45	-	-	-	
VoxNet [5]	92	0.9M	12 hours	6 ms	K40
OctNet [19]	88.03	-	-	-	
3D ShapeNets [6]	83.54	38M	48 hours	-	
VRN Ensemble [63]	**97.14**	90M	6 days	-	Titan X
SliceNet [8]	92.7	**0.24**	8.65 hours	3.7 ms	GTX1070/1920
SparseVoxNet (Ours)	92.7	1.5M	**36 min**	8.1ms	**GTX1050Ti**
SparseVoxNet with RF (Ours)	93.5	1.4M+	**38 min**	7.8ms	**GTX1050Ti**

Fig. 11: The confusion matrix describing the performance of a classification model on the ModelNet10 dataset. Our SparseVoxNet shows significant results just by preparing voxel data as its input. After the second training in Random Forest (RF), the accuracies are increased in most of the categories, as it is shown in the single raw at the bottom.

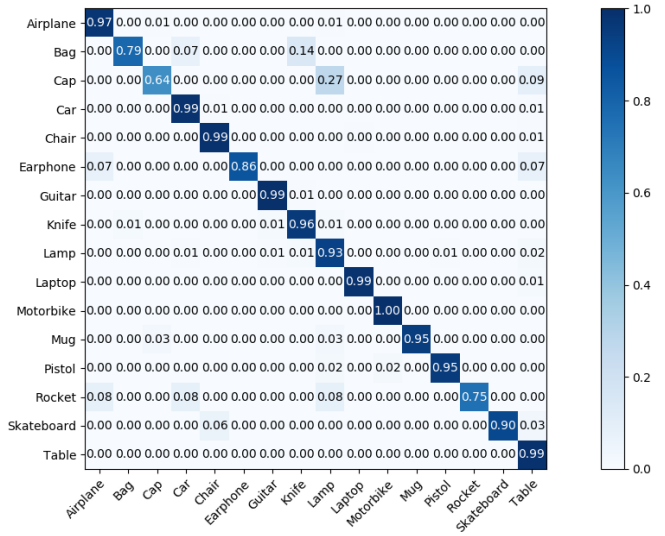


Fig. 12: The confusion matrix describing the performance of a classification model on the ShapeNetCore-part dataset. Our SparseVoxNet achieves this result with all the extra information, such as voxel normal and voxel curvature information.

are divided to train, test, and validation based on [37]. The result shows, in the designed model, increasing the grid size will increase the accuracy by having more details of the input model. However, when the grid size is increased, the more network's parameters are required to keep the efficiency of our method. Additionally, the training level would require a high-performance processor and much more memory by having a larger grid size. In this case, the proposed method shows notable performance with just 1.5M parameters, meanwhile a considerable number of input, which is 125,000 values for each input 3D object. The comparison of Fig. 10 depicts that the grid size with the value of 50 in all directions could be the most efficient options amoung the series of diffrent sizes such as 10, 20 , 25, 35, 45, 50, 55, 60. It does not require much memory, as the higher resolutions need, the training process is less time consuming, and it shows one of the highest performances, especially in comparison with the volumetric recognizer methods.

B. 3D Recognition

The comparison of our approach with state-of-the-art methods shows undeniable progress in 3D mesh recognition of different datasets. There are two main applications for the proposed method: recognizer and part annotator. Firstly, the estimation of voxel normal is fulfilled by training and predicting VoxNormNet as one of the preprocessing stages. Secondly, another essential feature is voxel curvature, which estimated by VoxCurvNet. These features allow us to have more information on the 3D object surface in a volumetric data format. Finally, for recognition purposes, the SparseVoxNet is provided. This well-structured network classifies the input 3D model through a deep and supervised neural network. By combining the preliminary networks' output to the third one, we achieve the best performance in recognition challenge according to the number of parameters we have used.

Table III presents the overall classification results of different methods on the ModelNet10 dataset. It shows that our merged network result is superior compared with state-of-the-art comparative methods when measured for its training speed, and it is also one of the top methods in terms of network size that make it easier to train in average GPU devices. The comparison demonstrates the power of combined normal and curvature data in the SparseVoxNet network for classification. In contrast to image-based techniques such as MVCNN [9], our approach does not require rendering of the model in different view angles; the only time-consuming task is prepro-

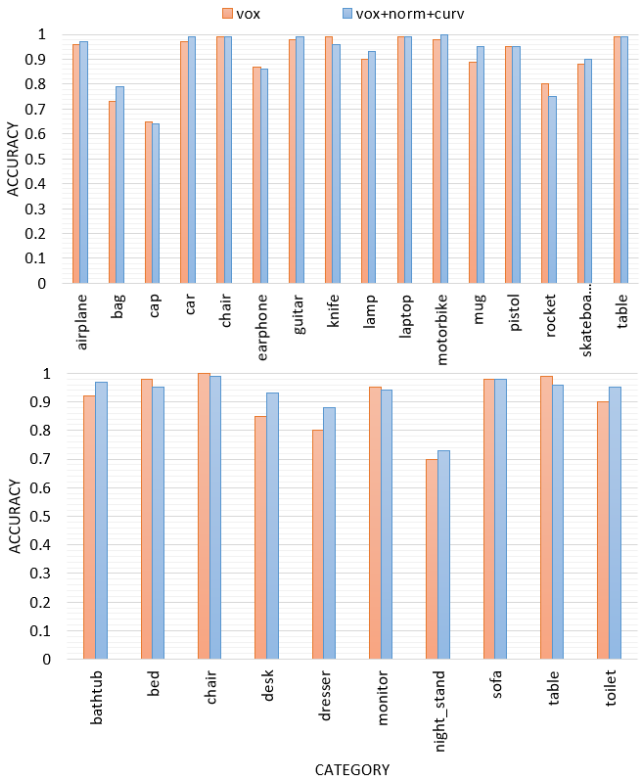


Fig. 13: Our recognition accuracy on two different datasets with and without surface features. The red bars represent recognition accuracy using just the voxels, and the blue bars represent the results using voxel, surface-normal, and surface-curvature. Top chart shows the results on ShapeNetCore-part and the bottom one is the result of Modelnet10.

cessing the data to extract voxel, normal, and curvature. This task is much faster than most other conventional methods by taking advantage of suggested VoxNormNet and VoxCurvNet. Training speed is also quicker on a similar challenge, which is due to employing a fewer number of parameters and a delicate network design. For our suggested structure, the SparseNet architecture has been leveraged to have fewer parameters in the network model while providing higher training efficiency.

Fig. 11 and Fig. 12 illustrates the confusion matrixes of the proposed deep neural network on Modelnet10 and ShapeNetCore-part, respectively. Note that ShapeNetCore-part is not designed as a test-bed for object recognition. However, because of the provided normal and curvature data, the method could draw advantage of all the abilities of the suggested network. Table IV demonstrates the results of the recognition using our method with a different combination of the inputs and features used. Table IV presents the progress of recognition accuracy after concatenating networks that take advantage of VoxNormNet and VoxCurvNet. The precise result of using volumetric data separately verifies the importance of raw volumetric data as our main entry, but by concatenating normal and curvature data, improved classification is achieved.

Fig. 13 illustrates the list of recognition accuracy of our method with and without extra input models. The results of the first bar of each category demonstrate the recognition accuracy

TABLE IV: The classification accuracy of our network by different combinations of data on the ShapeNetCore-part dataset. Using only voxel data in our method, achieved acceptable performance. But to increase the accuracy, the mixture of different preprocessed data with the input network is proposed.

Input Volumetric Data	Point's Normal	Curvature	Accuracy
✓			91.85%
	✓		74.70%
		✓	42.34%
✓	✓		92.48%
✓		✓	92.83%
	✓	✓	77.10%
✓	✓	✓	**93.21**%

TABLE V: Comparison of train, inference time, and GPU device used for the experiments of our approach with three other state-of-the-art methods. It shows the suggested method is faster than the point cloud based method in ModelNet40.

	inference time	training time	GPU
Ours	**7.8 ms**	**1h 55 min**	GTX 1050 Ti
PointCNN	12 ms	-	Tesla P100
PointConv	62ms	-	GTX 1080 Ti
KPConv	543 ms	134h 33 min	RTX 2080 Ti

based on the voxel input data, and the second one shows the precision with voxel normal and curvature data. The enhancement in the second bar is provided by the surface features, which are estimated by other networks. The best classification accuracy of SparseVoxNet on ModelNet10 is 92.7%. That is a significant improvement according to its number of network parameters (1.5M) compared to the pioneering work [5], [63]. There are also some drawbacks to the extra features that cause the method to achieve less accuracy in some of the objects. Here are our explanation of such problems:

- Firstly, the procedure of estimation of surface normal and curvature may not provide correct features for all the objects. This alone can prevent the training stage to get enough reached data and as the result we are losing the performance for some categories.
- Furthermore, considering all the provided features are accurate, the complexity of these features can alone cause this problem. The number of the extra data that we provided for the model entry gets very huge. Some of these extra data require more training time or more provided neurons to be learned that was not affordable for us to handle and it may cause the method to lose the efficiency.

Although we believed that using Modelnet40 is not fully matched with our future target dataset in medical school, we considered applying the suggested method on ModelNet40 to have a comparison with some of the state of the arts. In a couple of research in the past, people have researched using sparse data for processing 3D objects, especially in deep neural networks such as SparseConvNet [18]. However, based on our target dataset, which is provided in the School of Medicine, Shanghai Jiao Tong University, the 3D objects could be point clouds or volumetric data; we considered using volumetric data, as all other types could easily be converted. Therefore, sparsity comes from the connectivity between layers, which

TABLE VI: Segmentation accuracy comparison of our part-annotator network with state-of-the-art methods. Inference time and model size are added for those provided such information. We show the accuracy in different object categories separately.

	Voxel CNN	ACNN	Yi et al. 2016	Qi et al. 2016	Pointnet++ 2017	Yi et al. 2016	Ours
knife	79.58	81.98	85.4	85.9	85.9	86.1	**88.43**
aero	75.14	76.35	81	**83.4**	82.4	81.6	75.6
cap	73.28	70.8	77.7	82.5	87.7	81.9	**93.4**
lamp	74.43	77.43	82.5	80.8	83.7	84.7	73.31
guitar	88.35	87.84	92	91.5	91	93	**93.63**
mug	91.79	89.49	91.9	93	94.1	92.7	**96.51**
skate	65.25	82.05	69.8	72.8	76.4	82.9	**83.82**
rocket	51.16	49.23	53.1	57.9	58.7	60.6	**73.17**
pistol	76.41	77.41	85.9	81.2	81.3	81.6	**86.35**
laptop	93.92	95.49	95.7	95.3	95.3	95.6	**96.55**
earphone	63.5	71.14	61.9	73	71.8	74.9	**83.38**
motor	58.67	45.68	70.6	65.2	71.6	66.7	**77.69**
bag	72.8	72.89	78.4	78.7	79	81.7	**93.08**
car	70	72.72	75.7	74.9	77.3	75.2	**84.99**
chair	87.17	86.12	87.6	89.6	**90.8**	90.2	83.95
table	77.08	76.71	75.33	80.6	**82.6**	82.1	77.3
mean	74.76	75.77	79.28	80.39	81.8	81.96	**85.07**
Inference Time (s)	0.23	5	60	0.011	0.087	-	0.052
Model Size (MB)	3.75	-	-	9.4	12	-	4.02

increases the efficiency and accuracy of our method. Although 3D SparseConvNet has also provided a fast training procedure still our suggested method outperforms it to get 0.83 mean accuracy on ModelNet40 rather than 0.81 of SparseConvNet. We also considered ModelNet40 for comparison of training and testing time. We compare the training stage of our model with some of the point cloud based methods. Table V shows the result of our comparison, which explain that our efficient model using volumetric data can be much faster than point cloud methods on the ModelNet40 dataset.

C. 3D Segmentation

The main proposition in this paper is about finding an acceptable and accurate method for recognizing 3D models. However, by expanding the approach and applying almost the same procedure, just by modifying the final part of the suggested recognition network, the segmentation challenge can be explored. For comparison in this challenge, the ShapeNetCore-part dataset has been applied, which contains 16 different categories. As in the recognition structure, at the first stage, the parameters such as voxel normal and voxel curvature have to be prepared. This extra information will be merged into the voxel data and then feed the SparseVoxNet-part, which is the proposed network to untangle the segmentation challenge.

Table VI shows the results on 3D model segmentation on ShapeNetCore-part dataset. Our method acquired very precise results in comparison with some of the state of the arts such as PointNet++ [2], VoxNet [5], or SyncSpecCNN [38]. As the results indicate, we achieved the highest performance by a large margin and thus validating our method. We also added inference time to Table VI, making it comparable with other techniques for their process timing. Moreover, the Model size, which is in MB (Megabytes), is the network's size in GPU memory. The comparison of the results shows that the number of objects in each category is an important factor. If there are many different shapes in a category and it has variety in shapes, the accuracy will be increased. Conversely, if the number of objects is fewer, the network will not be able to segment the category as accurately as the first group. Therefore, the best network results are in categories that have an adequate number of objects. Fig. 14 demonstrates some samples of the segmentation result on ShapeNetCore-part. In our comparison, we found the results from our network to be able to capture better structures from the 3D models than that of other state-of-the-art methods. There are some results presented in Fig. 15(a) that shows a failure on the recognizer caused by the similarity of two different object classes. This also affect the segmentation in some object'c categories that intrigues us to plan to continue this research for further improvements.

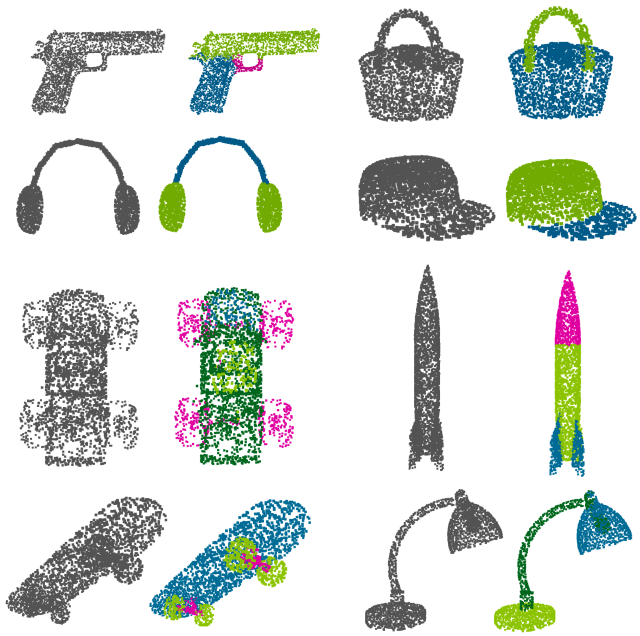


Fig. 14: Part annotation results in 6 categories of the ShapeNetCore-part. For each object pair, the left shows the original point cloud without segmentation, and the right point cloud shows the segmented counterpart.

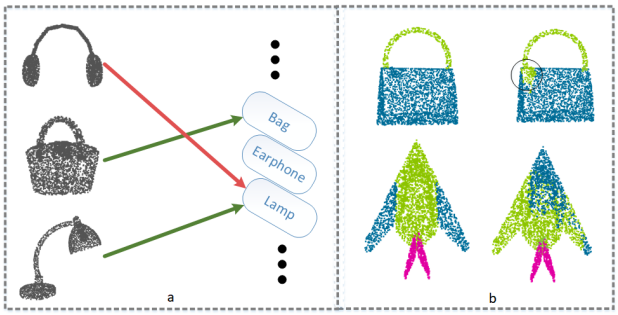


Fig. 15: (a) Some of the classification errors that happen between similar shape objects, (b) some samples of part annotation faulty result on the ShapeNetCore-part dataset.

VII. Conclusion and Future Work

In this paper, we have introduced a deep neural network which can classify 3D objects and verify it by statistical information from part annotator neural network. Our approach explores two essential features of the 3D meshes comprising of the surface normal and surface curvature; the first feature leverage directional variations while the second feature concentrates on changes through the point cloud's surface. Our experimental data suggest using surface features along with raw volumetric data to train rapidly and reach a high accuracy. Furthermore, employing Sparse connectivity among our convolutional layers, assist us to decrease the number of parameters required in the suggested network. We also described principle ways to define convolution operation on 3D input data and the process of concatenation in these 3D layers. Experimental results demonstrate that the proposed method is competitive with the state-of-the-art techniques on ModelNet10 and ShapeNetCore-part datasets. In the future, we will further employ segmentation results in the evaluation of the recognizer. If the recognition stage classified the model in the correct category, the segmentation stage would be statistically matched to the selected category. However, if the 3D model is classified as a wrong category, the segmentation of that object can be used to distinguish false recognition. We will further enhance the segmentation as our future work.

References

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 77–85.

[2] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Neural Information Processing Systems*, 2017, pp. 5099–5108.

[3] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3D semantic segmentation of point clouds," in *IEEE International Conference on Computer Vision Workshops*, 2017, pp. 716–724.

[4] S.-M. Hu, J.-X. Cai, and Y.-K. Lai, "Semantic labeling and instance segmentation of 3D point clouds using patch context analysis and multiscale processing," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–14, 2018.

[5] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 922–928.

[6] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[7] S. Zhi, Y. Liu, X. Li, and Y. Guo, "LightNet: A lightweight 3D convolutional neural network for real-time 3D object recognition," in *Eurographics Workshop on 3D Object Retrieval*, 2017, pp. 9–16.

[8] X. Chen, Y. Chen, K. Gupta, J. Zhou, and H. Najjaran, "SliceNet: A proficient model for real-time 3D shape-based recognition," *Neurocomputing*, vol. 316, pp. 144–155, 2018.

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

[10] T. Yu, J. Meng, and J. Yuan, "Multi-view harmonized bilinear network for 3D object recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 186–194.

[11] Q. Cui, S. Wu, Q. Liu, W. Zhong, and L. Wang, "MV-RNN: A multi-view recurrent neural network for sequential recommendation," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2018.

[12] H. You, Y. Feng, X. Zhao, C. Zou, R. Ji, and Y. Gao, "PVRNet: Point-view relation neural network for 3D shape recognition," in *AAAI Conference on Artificial Intelligence*, 2019, pp. 9119–9126.

[13] W. Guo and P. Aarabi, "Hair segmentation using heuristically-trained neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 1, pp. 25–36, 2018.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[15] S. Mohamad, A. Bouchachia, and M. Sayed-Mouchaweh, "A bi-criteria active learning algorithm for dynamic data streams," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 1, pp. 74–86, 2018.

[16] K. Ding, C. Huo, B. Fan, S. Xiang, and C. Pan, "In defense of locality-sensitive hashing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 1, pp. 87–103, 2018.

[17] H.-G. Han, W. Lu, Y. Hou, and J.-F. Qiao, "An adaptive-PSO-based self-organizing RBF neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 1, pp. 104–117, 2018.

[18] B. Graham, M. Engelcke, and L. v. d. Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9224–9232.

[19] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6620–6629.

[20] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-CNN: Octree-based convolutional neural networks for 3D shape analysis," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 72:1–72:11, 2017.

[21] L. Zhu, R. Deng, M. Maire, Z. Deng, G. Mori, and P. Tan, "Sparsely aggregated convolutional networks," in *European Conference on Computer Vision*, 2018, pp. 192–208.

[22] B. Athiwaratkun and K. Kang, "Feature representation in convolutional neural networks," *CoRR*, vol. abs/1507.02313, pp. 1–6, 2015.

[23] A. Boulch and R. Marlet, "Deep learning for robust normal estimation in unstructured point clouds," *Computer Graphics Forum*, vol. 35, no. 5, pp. 281–290, 2016.

[24] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "Nesti-Net: Normal estimation for unstructured 3D point clouds using convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1–9.

[25] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2019.

[26] W. Czajewski and K. Kołomyjec, "3D object detection and recognition for robotic grasping based on RGB-D images and global features," *Foundations of Computing and Decision Sciences*, vol. 42, no. 3, pp. 219–237, 2017.

[27] R. Socher, B. Huval, B. Bhat, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3D object classification," in *Neural Information Processing Systems*, 2012, pp. 656–664.

[28] R. B. Gomes, B. M. F. da Silva, L. K. de Medeiros Rocha, R. V. Aroca, L. C. P. R. Velho, and L. M. G. Gonçalves, "Efficient 3d object recognition using foveated point clouds," *Computers & Graphics*, vol. 37, no. 5, pp. 496–508, 2013.

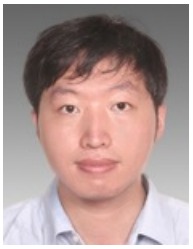
[29] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang, "Large-scale image classification: Fast feature extraction and SVM training," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1689–1696.

[30] Z. Yang, Y. Li, J. Yang, and J. Luo, "Action recognition with spatio-temporal visual attention on skeleton image sequences," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–11, 2018.

[31] C. Gan, N. Wang, Y. Yang, D.-Y. Yeung, and A. G. Hauptmann, "DevNet: A deep event network for multimedia event detection and evidence recounting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2568–2577.
[32] G. Gkioxari and J. Malik, "Finding action tubes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 759–768.
[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015, pp. 1–14.
[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "ImageNet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.
[35] C. R. Qi, H. Su, M. Niener, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5648–5656.
[36] V. Hegde and R. Zadeh, "FusionNet: 3D object classification using multiple data representations," in *3D Deep Learning Workshop at NIPS*, 2016, pp. 1–9.
[37] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A scalable active framework for region annotation in 3D shape collections," *ACM Transactions on Graphics*, vol. 35, no. 6, pp. 210:1–210:12, 2016.
[38] L. Yi, H. Su, X. Guo, and L. Guibas, "SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6584–6592.
[39] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.
[40] S. Zhi, Y. Liu, X. Li, and Y. Guo, "Toward real-time 3D object recognition: A lightweight volumetric CNN framework using multitask learning," *Computers & Graphics*, vol. 71, pp. 199–207, 2018.
[41] W. Liu and K. Zeng, "SparseNet: A sparse DenseNet for image classification," *CoRR*, vol. abs/1804.05340, pp. 1–17, 2018.
[42] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated CNN architecture design based on blocks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1242–1254, 2020.
[43] D. Liang, K. Weng, C. Wang, G. Liang, H. Chen, and X. Wu, "A 3D object recognition and pose estimation system using deep learning method," in *IEEE International Conference on Information Science and Technology*, 2014, pp. 401–404.
[44] H. Liu, Y. Cong, and Y. Tang, "Deep learning of volumetric representation for 3D object recognition," in *Youth Academic Annual Conference of Chinese Association of Automation*, 2017, pp. 663–668.
[45] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class hough forests for 3D object detection and pose estimation," in *European Conference on Computer Vision*, 2014, pp. 462–477.
[46] C. Ma, W. An, Y. Lei, and Y. Guo, "BV-CNNs: Binary volumetric convolutional networks for 3D object recognition," in *British Machine Vision Conference*, 2017, pp. 148:1–148:12.
[47] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Transactions on Graphics*, pp. 1–13, 2019.
[48] J. Xie, G. Dai, F. Zhu, L. Shao, and Y. Fang, "Deep nonlinear metric learning for 3-D shape retrieval," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 412–422, 2018.
[49] J. Gui, T. Liu, D. Tao, Z. Sun, and T. Tan, "Representative vector machines: A unified framework for classical classifiers," *IEEE Transactions on Cybernetics*, vol. 46, no. 8, pp. 1877–1888, 2016.
[50] D. Tao, X. Lin, L. Jin, and X. Li, "Principal component 2-D long short-term memory for font recognition on single chinese characters," *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp. 756–765, 2016.
[51] M. Sepahvand, F. Abdali-Mohammadi, and F. Mardukhi, "Evolutionary metric-learning-based recognition algorithm for online isolated Persian/Arabic characters, reconstructed using inertial pen signals," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2872–2884, 2017.
[52] Q. Feng, C. Yuan, J.-S. Pan, J.-F. Yang, Y.-T. Chou, Y. Zhou, and W. Li, "Superimposed sparse parameter classifiers for face recognition," *IEEE Transactions on Cybernetics*, vol. 47, no. 2, pp. 378–390, 2017.
[53] A. Majumder, L. Behera, and V. K. Subramanian, "Automatic facial expression recognition system using deep network-based data fusion," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 103–114, 2018.
[54] J. Zhang, G. Ma, Y. Huang, J. sun, F. Aslani, and B. Nener, "Modelling uniaxial compressive strength of lightweight self-compacting concrete using random forest regression," *Construction and Building Materials*, vol. 210, pp. 713–719, 2019.
[55] D. Liu, Z. Fan, Q. Fu, M. Li, M. A. Faiz, S. Ali, T. Li, L. Zhang, and M. I. Khan, "Random forest regression evaluation model of regional flood disaster resilience based on the whale optimization algorithm," *Journal of Cleaner Production*, vol. 250, pp. 119 468:1–119 468:15, 2020.
[56] M. Zhang, S. Khan, and H. Yan, "Deep eigen-filters for face recognition: Feature representation via unsupervised multi-structure filter learning," *Pattern Recognition*, vol. 100, pp. 107 176:1–107 176:14, 2020.
[57] Z. Xu, S. Li, J. Xu, J. Liu, X. Luo, Y. Zhang, T. Zhang, J. Keung, and Y. Tang, "LDFR: Learning deep feature representation for software defect prediction," *Journal of Systems and Software*, vol. 158, pp. 110 402:1–110 402:20, 2019.
[58] C. Ao, W. Zhou, L. Gao, B. Dong, and L. Yu, "Prediction of antioxidant proteins using hybrid feature representation method and random forest," *Genomics*, vol. 112, no. 6, pp. 4666–4674, 2020.
[59] R. B. Rusu, "Semantic 3D object maps for everyday manipulation in human living environments," *Künstliche Intelligenz*, vol. 24, no. 4, pp. 345–348, 2010.
[60] R. Atienza, "Fast disparity estimation using dense networks," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 3207–3212.
[61] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2261–2269.
[62] 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.
[63] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," in *3D Deep Learning Workshop at NIPS*, 2016, pp. 1–9.

Ahmad Karambakhsh received the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2020.

He is currently a Lead Computer Vision Scientist with the Ascent Solutions Pte Ltd, Singapore. He was a Computer Vision Developer with Zhitang, Shanghai, China. His current research interests include computer vision, graphics, and robotics.

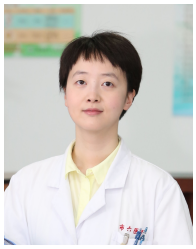


Bin Sheng (Member, IEEE) received the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Shatin, Hong Kong, in 2011.

He is currently a Full Professor with the Shanghai Jiao Tong University, Shanghai, China. He is an Associate Editor of the IEEE Transactions on Circuits and Systems for Video Technology. His current research interests include virtual reality and computer graphics.

Ping Li (Member, IEEE) received the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Shatin, Hong Kong, in 2013.

He is currently an Assistant Professor with The Hong Kong Polytechnic University, Kowloon, Hong Kong. He has published many top-tier scholarly research papers and has one excellent research project reported worldwide by *ACM TechNews* His current research interests include image/video stylization, artistic rendering and synthesis, and creative media.



Huating Li received the Ph.D. degree in internal medicine from Shanghai Jiao Tong University, Shanghai, China, and Pennington Biomedical Research Center, Baton Rouge, LA, USA, in 2011.

She is currently an Associate Professor with the Shanghai Jiao Tong University Affiliated Sixth People's Hospital and the Shanghai Diabetes Institute. Her current research interests include the role of cytokines in the development of fatty liver disease, diabetes, and other obesity-related diseases.

Jinman Kim (Member, IEEE) received the B.S. (Hons.) and Ph.D. degrees in computer science both from The University of Sydney, Sydney, Australia, in 2001 and 2006, respectively.

He is currently a Full Professor with the School of Computer Science, The University of Sydney, Sydney, Australia. His current research interests include medical image analysis and visualization, computer-aided diagnosis, and telehealth technologies.

Younhyun Jung received the B.Sc. degree in computer science from the Inha University, Incheon, Korea, in 2008, and the Ph.D. degree in computer science from The University of Sydney, Sydney, Australia, in 2016.

He is currently an Assistant Professor with the School of Computing, Gachon University, Seongnam, South Korea. He was a Research Fellow with The University of Sydney, Sydney, Australia. He worked at Samsung Electronics as a software engineer from 2007 to 2010. His current research interests are volume rendering, and multi-modal medical image visualization.

C. L. Philip Chen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1988.

He is currently a Chair Professor and the Dean of the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. Being a Program Evaluator of the Accreditation Board of Engineering and Technology Education (ABET) in the U.S., for computer engineering, electrical engineering, and software engineering programs, he successfully architects the University of Macau's Engineering and Computer Science programs receiving accreditations from Washington/Seoul Accord through Hong Kong Institute of Engineers (HKIE), of which is considered as his utmost contribution in engineering/computer science education for Macau as the former Dean of the Faculty of Science and Technology. He is a Fellow of IEEE, AAAS, IAPR, CAA, and HKIE; a member of Academia Europaea (AE), European Academy of Sciences and Arts (EASA), and International Academy of Systems and Cybernetics Science (IASCYS). He received IEEE Norbert Wiener Award in 2018 for his contribution in systems and cybernetics, and machine learnings. He is also a highly cited researcher by Clarivate Analytics in 2018 and 2019.

His current research interests include systems, cybernetics, and computational intelligence. Dr. Chen was a recipient of the 2016 Outstanding Electrical and Computer Engineers Award from his alma mater, Purdue University (in 1988), after he graduated from the University of Michigan at Ann Arbor, Ann Arbor, MI, USA in 1985. He was the IEEE Systems, Man, and Cybernetics Society President from 2012 to 2013, the Editor-in-Chief of the IEEE Transactions on Cybernetics (2020-2021) and the IEEE Transactions on Systems, Man, and Cybernetics: Systems (2014-2019), and currently, an Associate Editor of the IEEE Transactions on Fuzzy Systems. He was the Chair of TC 9.1 Economic and Business Systems of International Federation of Automatic Control from 2015 to 2017, and currently is a Vice President of Chinese Association of Automation (CAA).