Received August 23, 2020, accepted August 31, 2020, date of publication September 3, 2020, date of current version September 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3021455

A Voxelized Fractal Descriptor for 3D Object Recognition

JOSE FRANCISCO DOMENECH, FELIX ESCALONA, FRANCISCO GOMEZ-DONOSO, AND MIGUEL CAZORLA, (Senior Member, IEEE)
University Institute for Computer Research, 03080 Alicante, Spain

Corresponding author: Felix Escalona (felix.escalona@ua.es)

This work was supported in part by the Spanish Government, with Feder funds, under Grant PID2019-104818RB-I00, and in part by the Spanish Grants for Ph.D. studies under Grant ACIF/2017/243 and Grant FPU16/00887.

ABSTRACT Currently, state-of-the-art methods for 3D object recognition rely in a deep learning-pipeline. Nonetheless, these methods require a large amount of data that is not easy to obtain. In addition to that, the majority of them exploit features of the datasets, like the fact of being CAD models to create rendered representation which will not work in real life because the 3D sensors provide point clouds. We propose a novel global descriptor for point clouds which takes advantage of the fractal dimension of the objects. Our approach introduces many benefits, such as being agnostic to the density of points of the sample, number of points in the input cloud, sensor of choice, and noise up to a level, and it works on real life point cloud data provided by commercial sensors. We tested our descriptor for 3D object recognition using ModelNet, which is a well-known dataset for that task. Our approach achieves 92.84% accuracy on the ModelNet10, and 88.74% accuracy on the ModelNet40.

INDEX TERMS 3D object recognition, fractal, global descriptor, machine learning.

I. INTRODUCTION

The ability to recognize tridimensional objects is a key feature for a range of different applications. For instance, intelligent and self-driving cars must be able to detect and recognize the objects that surround them in order to enable a specific behaviour: they must stop in a red light, they must engage the emergency brake if a pedestrian suddenly crossed the road or they must recognize the traffic signs to drive safely.

Another notorious application for the tridimensional object recognition could be social robotics. The robots that are meant to interact with humans should be able to recognize each of them, and also to recognize certain objects that may be potentially involved in their tasks. For instance, it might be useful for the robot to recognize household items.

So far, the most novel and accurate 3D object detectors rely in a deep learning pipeline. However, these methods require a large amount of annotated and structured data in order to be trained, which is very hard to obtain. Due to this, most of the deep learning-based 3D object detectors do not perform that well in real life, off-lab tests. In addition to that, most of them are coupled to a certain kind of sensor. For instance, there are approaches that only work on 16-Beams LiDAR data, or they only work on 3D Computer Aided Designed (CAD) objects.

In this article, we propose a novel global descriptor, which is based on the fractal dimension. The voxel-based computation of the fractal dimension is agnostic to the density of points, number of points in the input cloud, sensor of choice, and noise up to a level. In addition to that, it is very fast to compute. The fractal descriptor could be used for 3D object recognition, which is the feature that is most explored in this work.

Thus, the main contribution of this work is a novel global descriptor that is based on the computation of the fractal dimension, which we called Voxelized Fractal Descriptor (VFD).

The rest of the paper is organized as follows: first, we review some relevant related works in Sec. II. Then, we describe our approach in Sec. III. In Sec. IV, we show the results of the experiments we carried out to validate our proposal. Finally, we draw the conclusions and state the limitations and future works in Sec. V.

II. RELATED WORKS

Object recognition is a very important topic in the research community, as it can provide the understanding of the scene

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaohui Yuan.

for an artificial intelligence. At the beginning of computer science, the main focus was in object recognition with images. Deep learning approaches achieved high accuracy levels in many competitions [1], even better than humans, with *Convolutional Neural Network* architectures [2]. However, 2D object recognition relies on appearance and not the shape so, for example, it cannot differentiate between a picture and the object that has been depicted inside. With the advent of depth sensors, we can have information about the shape of the object, so we can differentiate between these classes. Nevertheless, the inherited unorganized representation of the 3D information, where the neighborhood of every point cannot be directly extracted, makes this task harder than the 2D counterpart [3].

Traditionally, 3D object recognition has been carried out with local and global feature-based descriptors and geometric primitives [4], [5].

On one side, local methods describe a single point and are only based on local geometric features around the points of interest. However, they require methods to select which points are in fact keypoints, and specialized classifiers to deal with multiple feature vectors per object [6].

On the other side, global methods describe the whole object with a single vector, which is directly suitable for standard, traditional classifiers. Nonetheless, they need a segmentation stage in order to isolate the object [7]. We consider that global techniques are more interesting than local approaches to be used as benchmark for further novel methods, as they can be directly used with the classifier of choice, so a revision of the most notorious alternatives of this family is provided in the following paragraphs.

Ensemble of Shape Functions (ESF) [8] is the only one descriptor that do not rely on normals to describe the shape. It is a combination of ten histograms generated from three different shape functions: point distance (D2, line between point-pairs randomly sampled), angle (A3, angle formed by 3 randomly sampled points), and area (D3, area created by three points) [9]. The input cloud representing the object is voxelized to approximate the surface and is used to trace the line joining pair of points efficiently. For every function, there are 3 histograms depending on the connectivity of the points that lies on the surface, off the surface or mixed, according to [10]. Finally, the distance ratio histogram is create using the lines from D2 function.

Viewpoint Feature Histogram (VFH) [11] is a global descriptor based on the *Fast Point Feature Histogram (FPFH)* local descriptor [12]. It is composed of two elements: a viewpoint direction component and an extended FPFH component. For the viewpoint direction, the centroid of the object is calculated, and the vector between the viewpoint of the camera and the centroid is then computed. Then, for every point in the object the angle between this vector and its normal is computed. The extended FPFH is computed as the original method, using the centroid as the keypoint and the rest of the points as its neighbors.

Clustered Viewpoint Feature Histogram (CVFH) [13] is an extension of the *VFH* descriptor that calculates a histogram for each region of points, and not for the whole object. These regions are split by removing points with a high curvature value, and then applying a region growing segmentation algorithm.

Finally, *Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram (OUR-CVFH)* [14] is an iteration of the *CVFH* method that defines a robust unique and repeatable reference frame to describe the object that allows avoiding the ambiguity over the camera roll angle.

With the explosion of new Artificial Intelligence (AI) hardware accelerators, many researchers are focusing on deep learning methods for tackling the 3D object recognition task. We introduce these methods by grouping them in three big categories according to their 3D data representations.

Point Cloud. 3D data is represented as a raw unordered point cloud. These methods usually extract features by analyzing the neighborhood of every point within a radius. The most representative proposal in this case is *PointNet++* [15]. It generates a feature vector for the whole cloud by applying order-invariant transformations to every point, generating local hierarchical features that are sampled and grouped, and uses it to segment and classify the scene.

Some proposals are based on the previous architecture. This is the case of *VoteNet* [16], a novel technique based on Hough voting, that uses *PointNet++* layers as the backbone. This approach selects a set of interesting points, with their corresponding features, as seed points to generate clusters of object instances based on their votes. Finally, these clusters are transformed into 3D bounding boxes with their corresponding categories.

Another work, *SplatNet* [17], extends the concept of 2D SPLAT images into 3D. It uses hash tables as a efficient implementation of neighborhood filtering, which provides an easy mapping of 2D points into 3D space, Then, bilateral convolutions are used to extract a set of features.

In the case of *SO-Net* [18], the authors propose a method to guarantee invariance to point permutations. It builds a Self-Organizing Map (SOM) through modelling the spatial distribution of the point cloud and using the neighborhood of every point to extract hierarchical features. As a final step, this method generates a global feature vector for the whole cloud.

Other approaches, like *Point-Voxel CNN (PVCNN)* [19], combine the sparse representation of the data with voxelized convolutions that increase the performance of the data access and improve the locality of the method. In this work, a new efficient primitive is introduced, Point-Voxel Convolution (PVConv), that converts points into voxel grids, aggregates neighboring points with voxel-based convolutions and transforms them back to points. In order to obtain features with a higher level of detail, they include point-based feature transformations.

2D projections. In this category of methods, input data is represented as multiple 2D projections of the tridimensional data. Traditionally, they have been the most common approaches, and usually have a 2D *Convolutional Neural Network* to carry out the processing, as presented in [20]. Some studies train a boosting classifier to group different views and improve the quality of the inference, like [21]. On the other hand, another works make a manual choice of the best views to perform the classification. This is the case of [22], that considers 3 orthogonal views to be good enough to carry out the classification task, and feed 3 independent neural networks, whose results are finally combined. Based on ModelNet benchmark, [23], these methods have the highest classification performances, but they need multiple views of the object, so they would perform poorly in real-world applications if they have to deal with with occlusions and partial views.

Voxelization. In this case, input data is a discretization of the original point cloud, grouping points into different clusters according to a neighborhood criteria, that serve as a approximation of the original shape. Commonly, every voxel is represented as a binary value, 0 or 1, that indicates the presence of points in the space the voxel represents. *3D ShapeNets* [24], the study carried out by the authors of ModelNet, discretizes the data as a cubic voxel and applies 3D convolutions to generate the features. In a similar manner, *VoxNet* [25] applies a 3D *Convolutional Neural Network* to the volumetric representation to generate the classification. Another works that are based on 3D convolutions are [26], [27], which also use the voxel representation of the objects. On another note, *PointGrid* [28] creates grid cells with a constant number of points with a point quantization technique, saving points' coordinates to improve the representation of the local geometry of the object. Other works, such as *O-CNN* [29] and *OGN* [30] combine the use of octree representations with the performance of 3D convolutions to lower memory consumption and improve the performance.

Similar to approaching the problem by using multiview 2D projections, works like *MO-VCNN* [31] generate different rotations of the 3D models and extract high level features from every pose that are combined into a final feature vector. Despite the popularity of 3D convolutions, other deep learning architectures have been used. *Vconv-dae* [32] uses a convolutional denoising auto-encoder as a feature learning network, [33] a *Variational auto-encoder* and [34] a *Generative Adversarial Network*.

The main problem of this category of techniques is the loss of detail due to the discretization step, and that require high amounts of memory for the 3D convolutions, which are mainly useless due to the internal void of the representation, namely, that the point cloud only has points in the surface of the object.

As for **fractals**, the fractal dimension has been tested as descriptor in the past, with promising results. Regarding the field of plant recognition, in [35] authors researched a range of algorithms for calculating the fractal dimension. They segment the veins of the plants and their borders, and they calculate the fractal dimension using this information, and classify them into different species of Passiflora, obtaining an average accuracy of 97,77%.

Another example of the use of the fractal dimension as descriptor is [36]. In this work, which is more theoretical, authors propose the box counting algorithm to calculate the fractal dimension. They aim to demonstrate how fractal geometry can describe properties of the urban development that cannot be capture with euclidean geometry. In this article, they use 2D binary images of maps of 20 different American cities to carry out the analysis. In this study, they conclude that there is a correlation between fractal dimension and the level of urban development of each city, but they found no evidence of a correlation between this dimension and the population density.

As a last example, we can find the article [37], which uses fractal dimension to analyse signals from a magnetoencephalogram. In this case, fractal dimension is considered as an indicator of the complexity of a signal, that is usually calculated following the Higuchi algorithm [38]. Authors seek to verify the reliability of the fractal dimension as an Alzheimer detector. Using this technique, they achieve a 87,8% of accuracy and a 95,24% of specificity, with 5 channels, so they obtained good results for diagnosing this disease.

There is a specific feature that appear in each of the methods described above, and that is the fact that all of them work with the fractal dimension in two dimensions. Because of this, in this work we propose a method to carry out 3D object recognition, taking advantage of the properties of the fractals, in the hope of popularizing its use.

III. PROPOSAL

In order to evaluate the feasibility of the fractal dimension as a 3D object descriptor, we propose the *Voxelized Fractal Descriptor* (VFD). The process to generate the VFD from an input point cloud is as follows: first, a point cloud is obtained. The points are not required to be sorted by any mean. A voxel grid clustering of an specific size is computed using the point cloud as an input. The resolution is a parameter of the approach. As a result, there are generated a set of clusters which contain the points in the input point cloud. Then, the fractal dimension is computed for each cluster. As a result, we obtain a set of fractal dimensions, one per cluster. If a cluster is empty, namely it has no points within, a fixed sentinel value of 0 is returned, corresponding with the theoretical fractal dimension of the empty set. Finally, the fractal dimensions are concatenated to generate the final voxelized fractal descriptor. As expected, the computational complexity of the descriptor is related to the resolution of the voxel grid filter, as more voxels implies the computation of more fractal dimensions, thus generating a larger descriptor. The process to generate the VFD is shown in Figure 1.

The explanation of the algorithm to calculate the fractal dimension is in Sec. III-A, and the details of our descriptor are in Sec. III-B

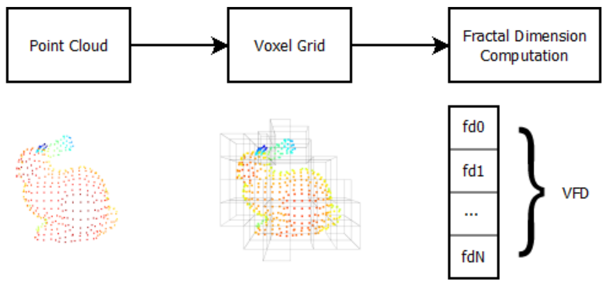


FIGURE 1. The process to generate a VFD from an input point cloud.

A. 3D FRACTAL DIMENSION ALGORITHM

The concept of fractal dimension, in fractal geometry, is a generalization of the concept of dimension in classic geometry. As a simplification, it can be defined as a measure of how much an object appears to fill the space as it is scaled up or down. A clear example is the Koch snowflake. This figure has a topological dimension of 1 but cannot be treated as a curve, because as we zoom in on the figure, the distance between its points tends to infinite. This phenomena is know as Coastline Paradox [39].

There are many ways to calculate this property (information dimension, correlation dimension, Higuchi dimension, Lyapunov dimension, Hausdorff dimension…), but we have chosen the box counting method, also known as Minkowski–Bouligand dimension. This method is one of the most popular to calculate the fractal dimension of sets in euclidean space, it is directly applicable to 3D data, and is very fast to compute.

The name of the box counting method is very descriptive, as it consists in dividing the space in boxes, and counting the number of cells that contain points. In more detail, the space occupied by the object is divided into square boxes of side size ε. Then, the cells that are occupied by the object are counted. This process is repeated *nIters* times with a fixed *boxSizeDecrease* decreasing value of ε, obtaining the Eq. 1, where $N(\varepsilon)$ is the number of boxes required to cover the figure according to the side size, ε is the side size, and D_{BC} is the box counting dimension.

$$N(\varepsilon) = \varepsilon^{D_{BC}} \tag{1}$$

The obtained expression is similar to those of Haussdorf and Higuchi, and can be operated with algorithms to generate a linear function, as shown in Eq. 2. As we can see, D_{BC} is the slope of the linear function. In our practical application, we will apply a least-squares adjustment to approximate the value of this slope, as proposed in [40].

$$log(N(\varepsilon)) = D_{BC} \cdot log(\varepsilon) \tag{2}$$

The translation of this algorithm to the 3D euclidean space is a trivial process, as the only difference is the substitution of the squares boxes of the grid by cubic voxels. To do so, we relied in the voxel grid algorithm. The only parameter to be set is the number of iterations *nIters* (decreases in box size), as the rest of parameters are calculated from this. Assuming that the object is centered in the origin, the initial box size, tam_0 is the distance from the origin to the furthest point of the object, and the decrease in box size is calculated following the Eq. 3

$$boxSizeDecrease = \frac{tam_0}{nIters} \tag{3}$$

The voxel grid filter is applied in each iteration, as shown in Fig. 2 and Fig. 3 with the corresponding size. Then, the number of voxels that are occupied is computed. We stored the corresponding values of voxel size and number of occupied cells in every iteration, and applied the Eq. 2 approximating, thus, the slope of the line with a least-squares adjustment. Finally, this value represents the fractal dimension, as shown in Fig. 4.

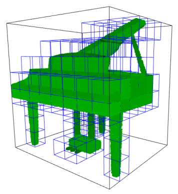
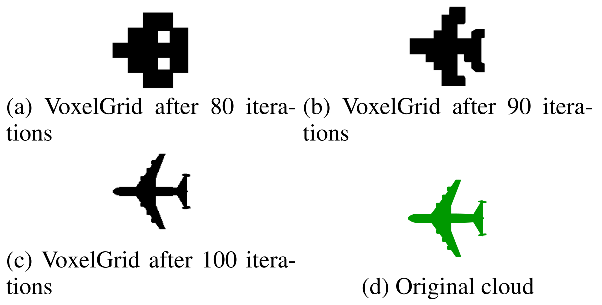


FIGURE 2. Sample of the division (in blue) of an instance of the *piano* class from *ModelNet40*. In black we can see the main bounding box of the figure, that marks the size for the divisions.



(a) VoxelGrid after 80 iterations (b) VoxelGrid after 90 iterations

(c) VoxelGrid after 100 iterations (d) Original cloud

FIGURE 3. Effects of the evolution of Voxel Grid filter in box counting.

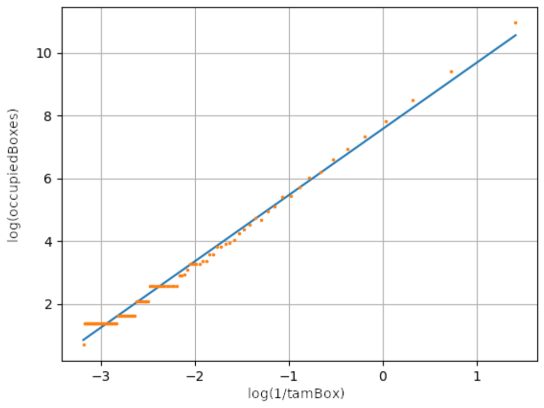


FIGURE 4. Example of least-squares approximation for fractal dimension.

B. 3D VOXELIZED FRACTAL DESCRIPTOR

The one-dimensional version of the descriptor is explained first. Given a 3D object defined by a set of 3D points that represent its surface, this descriptor is calculated for the whole object using the method explained in the previous section. Thus, as a result, there is a single value which is used as a global descriptor for the object. In Sec. IV we describe an experiment to check the efficacy of this method.

The next idea was to involve the distribution of points and their location on the object as well. With this in mind, our proposal is to divide the object in voxels, with a voxel resolution provided by the user. Then, the fractal dimension for the points that lie within each voxel is computed. If a voxel has no points within, a fractal dimension of 0 is returned. The fractal dimensions are concatenated in a single vector by following the order of creation, thus generating the final descriptor. We named this version the fractal Voxelized Fractal Descriptor (VFD). It is worth to mention that the only parameter that takes VFD is the resolution of the voxels. The descriptors of a set of objects are then used to train a classifier in order to tackle the 3D object recognition task.

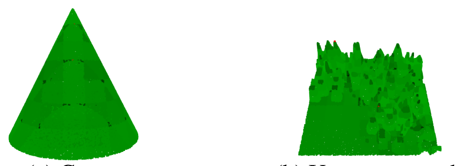


FIGURE 5. Plot of the calculation of fractal dimension over two voxelized figures, a cone and a ground with two different parts: a flat surface and a very sharp surface, with the lighter colours being higher values of fractal dimension.

Figure 5a shows the result of dividing a cone in voxels and calculating the fractal dimension for every voxel. The points that lie in a certain voxel are colored as a function of the fractal dimension. The voxels with the lightest shades of greens, which represent a higher fractal dimension, correspond with the zones with more points. The darkest areas represent a lower fractal dimension, which also are zones with fewer points. On the other side, in Fig. 5b, we can see the difference between a flat surface and a rough surface. Logically, for the regular part of the surface, the fractal dimension is constant, whilst the roughest parts show a range of different factal dimension. In this part, the bulge tips have low values, the points near the base have similar values to the flat surface, and the central parts present the highest values of fractal dimension, which are represented by the different shades of green.

In Fig. 6 we present the analysis for 4 different classes of the ModelNet40 dataset. Figure 6a shows a guitar, where its body and the central part of the neck are constant, because they are flat surfaces. The parts that show more difference are the edges of the guitar neck and body. Besides, in Fig. 6b, there is a darker band due to the intersection of some voxels with the limits of the cylinder. More dark points are on the

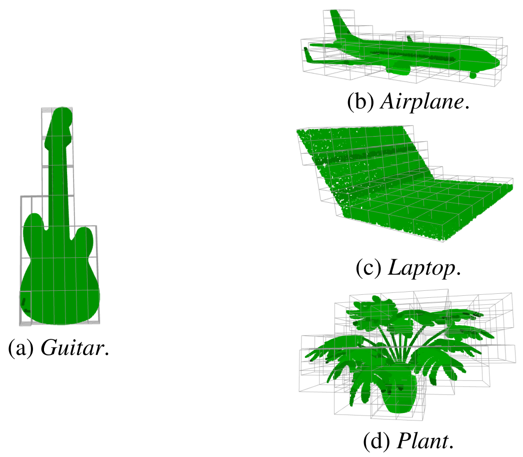


FIGURE 6. ModelNet40 samples of calculating fractal descriptor over voxels. Lighter green values are higher fractal dimensions.

winglets, at the end of the plane. The case of the laptop in Fig. 6c is pretty interesting because the keyboard is in constant color as it is vertically aligned, but the screen shows some discontinuities due to the fact that it is tilted, thus its points lie within multiple columns of the grid. Finally, in Fig. 6d we have the example of a plant, which leaf surfaces and vase are of similar, light green color, whilst the stem and leaf tips are of darker colors. This effect is identical to the one with the wings of the plane, the guitar tuning pegs and the tips of the rough surface. We strongly believe that this is an important factor to differentiate between objects, as we demonstrate in the next section.

IV. EXPERIMENTATION

In this section, we first describe the dataset, ModelNet, used in the experimentation. Then, we apply the one-dimensional descriptor to the smallest version of the dataset, ModelNet 10, and show the results in Sec. IV-B. After that, in Sec. IV-C, we present the results for the *ModelNet10* dataset for the Voxelized Fractal Descriptor. For this dataset, we conducted a great deal of tests in order to find the classifier and the voxel resolution that provide the best performance for our descriptor. Then, in Sec. IV-D we use the best setup found in the previous section with the most complex version of the dataset, which is *ModelNet40*. We do this to evaluate the performance of our system with harder problems, also with the VFD descriptor.

The experiments were carried out using the following setup: Intel Core i5-3570 with 16 GiB of Kingston HyperX 1600 MHz and CL10 DDR3 RAM on an Asus P8H77-M PRO motherboard (Intel H77 chipset). The implementation of our methods has been made with Python, using the novel Open3D library [41], which offers a large set of functions to process 3D data.

A. ModelNet DATASET

We have used the *Princeton ModelNet* project [24], which is one of the most widely used benchmarks for 3D object recognition. This dataset has two versions: ModelNet10 and ModelNet40.

ModelNet10 offers a set of more than 4,700 CAD models from 10 different categories that are manually aligned, and divided into training and test set. Following the steps explained in [26], we converted these models into *Point Cloud Data* (PCD) clouds, compatible with the *Open3D* library. In [26] we can also see the the highly imbalanced distribution of both training and test sets.

The main problem of this dataset is the visual similarity between categories, that mainly occurs with the *night_stand* y *dresser* categories. In Fig. 7 we can see samples of both categories, which are very difficult, even for a human, to distinguish them. It is clear that there are other designs which are more differentiable from each other, but this problem occurs quite often, thus leading to a potential decrease of the accuracy of the classification results.

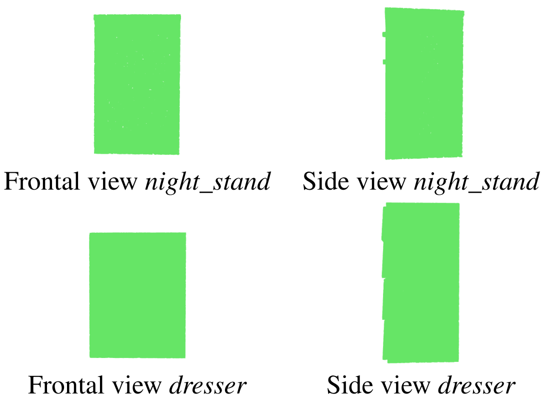


FIGURE 7. Comparative between instances of *night_stand* and *dresser* in frontal and side view.

ModelNet40 offers a set of more than 11,000 CAD models from 40 different categories manually aligned, and divided into training and test set. Similarly to the ModelNet10 dataset, we converted these models into PCD clouds. This version of the dataset also shows a high imbalanced distribution of both training and test sets.

The problem referred to ModelNet10 happens with this dataset too, as shown in Figures 8 and 9, as it is of the same nature. In addition to those already mentioned in the previous section, we will see frequent confusions between *door* and *curtain* classes, because they are both plain rectangles, taller than wide and shallow. Something similar occurs between *bench* and *desk*, because most of their instances consists of a large rectangular base with four legs. However, main confusions are between classes *flower_pot*, *plant*, *vase* and *cup*. We can find many potted plants in *flower_pot* and *plant* instances indistinctly, so the classification is expected to be somewhat arbitrary in this case. On the other hand, *flower_pot*, *vase* and *cup* categories have very similar instances that only differ in scale, color and use, feature that are not taken into consideration in the proposed VFD algorithm.

B. RESULTS ON ModelNet10 USING THE ONE-DIMENSIONAL DESCRIPTOR

The first experiment we carried out consists of a naive experiment, using the one-dimensional descriptor, which is described in Sec. III-B, for the ModelNet10 dataset. We have used the *K-Nearest Neighbors* (KNN) classifier, with a K value of 5. It showed a very poor accuracy of 21,6%, as shown in Fig. 10. This result is expectable as it is very likely that a range of different object share the same fractal dimension for the whole object. So, we conclude that this descriptor does not provide enough discriminative power to allow distinguish between categories.

C. RESULTS ON ModelNet10 USING VFD

The first classifier we tested was KNN. We tried different setups of k and voxel resolutions. The results are presented in Fig. 11. in this plot, it can be seen the influence of the k parameter and the voxel resolution. With the minimum number of subdivisions (2^3) the worst results appear, but they improve considerably from 3^3 to 10^3, where the accuracy starts decreasing slightly. Apparently, this algorithm works better with lower values of k, regardless of the voxel resolution. Moreover, it is worth noting that the number of

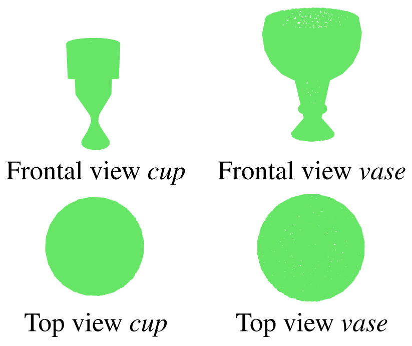


FIGURE 8. Comparative between instances of *cup* and *vase* in frontal and top view.

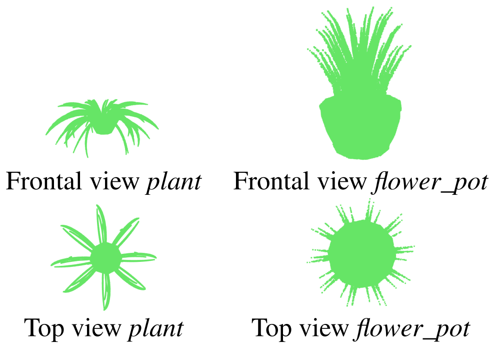


FIGURE 9. Comparative between instances of *plant* and *flower_pot* in frontal and top view.

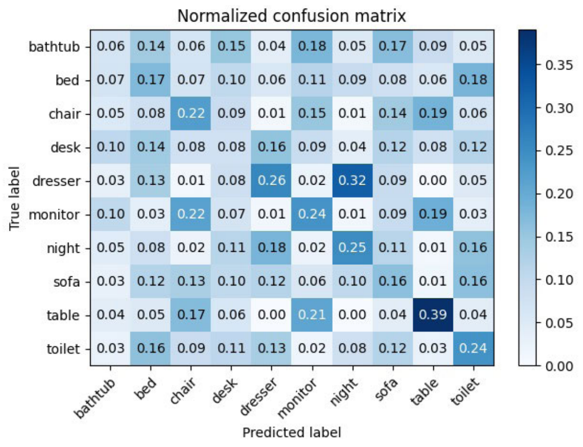


FIGURE 10. Confusion matrix with the one-dimensional fractal descriptor for the whole object with a KNN classifier, $k = 5$.

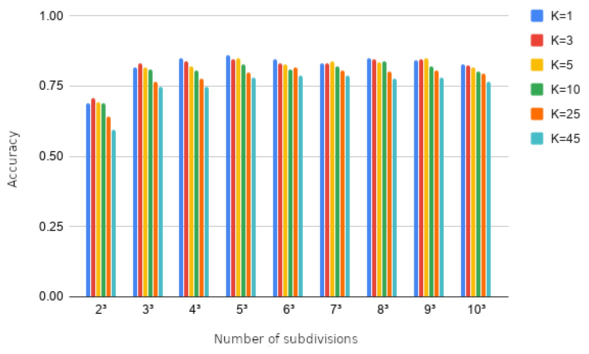


FIGURE 11. Comparison of ModelNet10 results for KNN with different k values and voxel subdivisions using euclidean distance.

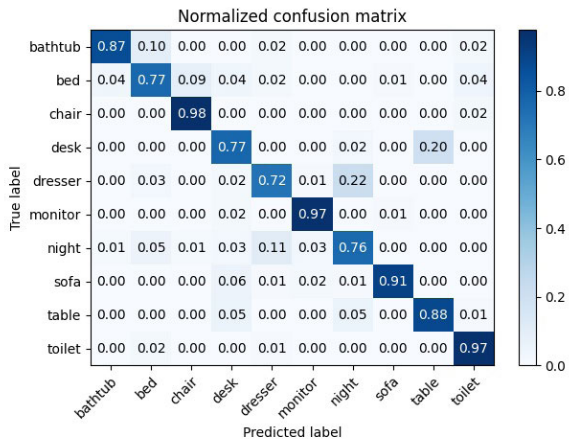


FIGURE 12. Confusion matrix for ModelNet10 using KNN with $k = 1$ and 5^3 subdivisions. 86,01% accuracy.

subdivisions with better average result is 5^3. Finally, the best result of a 86,01% accuracy is obtained with the configuration $k = 1$ and 5^3 subdivisions. The confusion matrix regarding this setup is shown in Fig. 12. This last result will be used to compare with other classifiers.

The next classifier we tested was a fully-connected neural network. We tried a simple topology as a baseline in order to know the reliability of our descriptor and prevent overfitting. This first architecture has only two layers: an input layer with a number of neurons equal to the size of the feature vector (that depends on the voxel resolution) with sigmoid activation, followed by an output layer of 10 neurons (one for each category) with softmax activation. The best results were obtained with 9^3 subdivisions, which yielded an accuracy of **90,75%**. The corresponding confusion matrix is shown in Fig. 13.

After observing the learning curves, we noticed a little overfitting, even though it is a very simple network, so we used regularization techniques such as Dropout [42]. This technique is very simple and consists of randomly disabling the connection between some neurons, so we force every neuron to learn something that contributes to the overall result of the network, preventing memorizing. After involving this technique, we achieved an accuracy of **90,97%**, but differences between train and test accuracy were significantly lowered, as depicted in Figs. 14, 15. 16. 17.

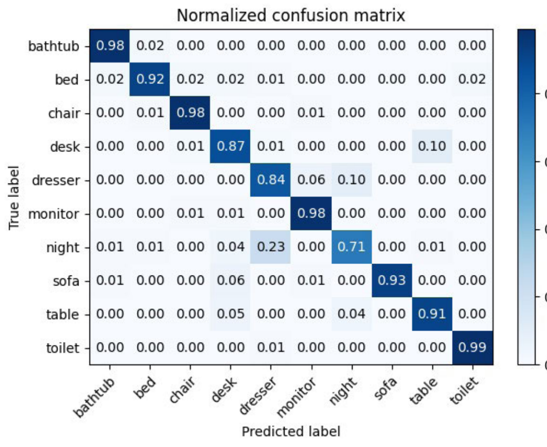


FIGURE 13. Confusion matrix for ModelNet10 with a fully-connected network and 9^3 voxel resolution. 90,75% accuracy.

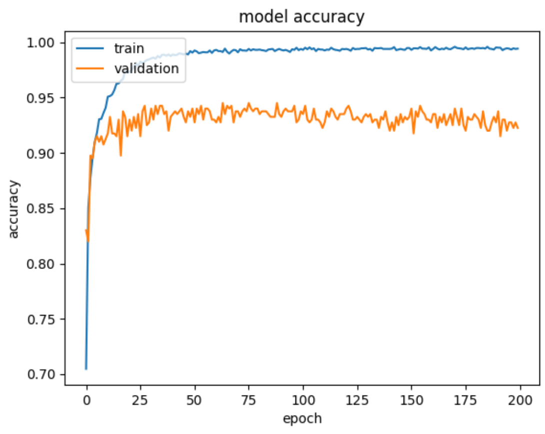


FIGURE 14. Accuracy before applying Dropout.

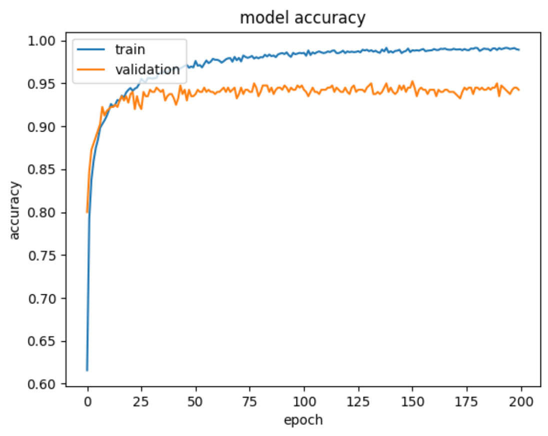


FIGURE 15. Accuracy after applying Dropout.

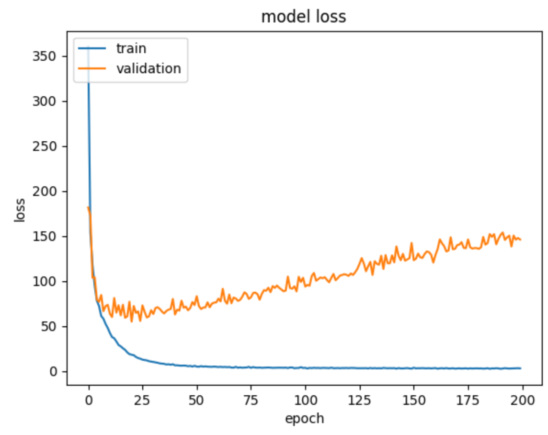


FIGURE 16. Loss before applying Dropout.

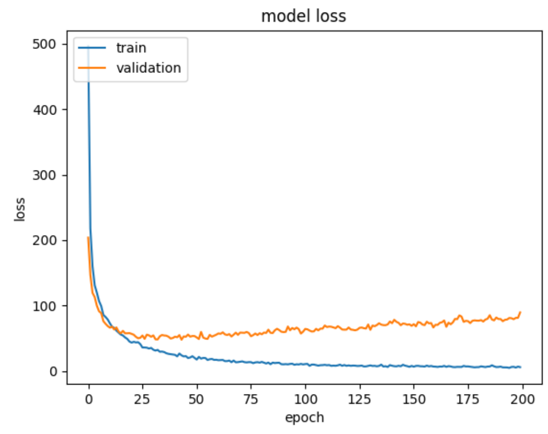
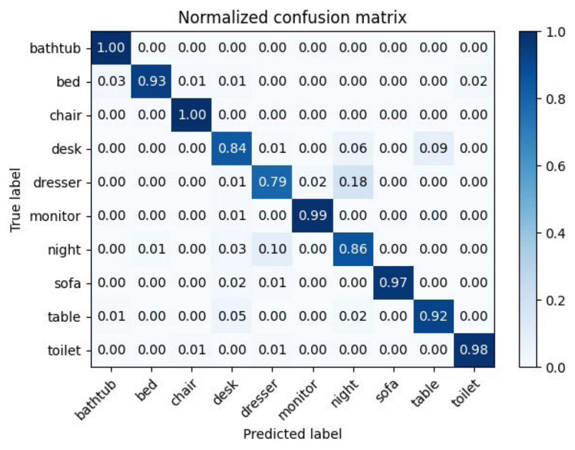


FIGURE 17. Loss after applying Dropout.



True label	bathtub	bed	chair	desk	dresser	monitor	night	sofa	table	toilet
bathtub	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bed	0.03	0.93	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.02
chair	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
desk	0.00	0.00	0.00	0.84	0.01	0.00	0.06	0.00	0.09	0.00
dresser	0.00	0.00	0.00	0.01	0.79	0.02	0.18	0.00	0.00	0.00
monitor	0.00	0.00	0.00	0.01	0.00	0.99	0.00	0.00	0.00	0.00
night	0.00	0.01	0.00	0.03	0.10	0.00	0.86	0.00	0.00	0.00
sofa	0.00	0.00	0.00	0.02	0.01	0.00	0.00	0.97	0.00	0.00
table	0.01	0.00	0.00	0.05	0.00	0.00	0.02	0.00	0.92	0.00
toilet	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.98

FIGURE 18. Confusion Matrix for ModelNet10 with SVM and 5^3 subdivisions. 92,84% accuracy.

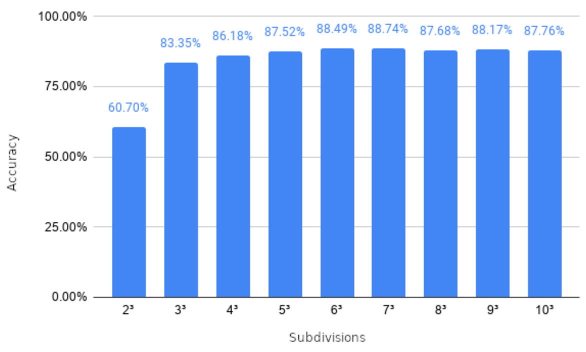


FIGURE 19. Accuracy for SVM classifier in *ModelNet40* with different voxel resolutions.

The last classifier we tested was *Support Vector Machines* (SVM). With this method we achieved the highest accuracy, **92,84%**, with 5^3 subdivisions. In the light of the results, we can determine that SVM is the best classifier for our descriptor.

Fig. 18 shows the confusion matrix regarding this experiment, where we can notice that a great part of the errors are misclassifications between conflicting classes, as exposed in Sec. IV-A. This fact shows that our system is working properly, as the confusions are between classes that are visually and semantically similar, which is very hard to tell apart even for humans.

D. RESULTS ON ModelNet40 USING VFD

In view of the success of the SVM classifier for *ModelNet10*, we decided to move on to the extended version of the dataset that involces 40 different classes. We carried out different experiments with a range of voxel resolutions, as shown in Fig. 19. Best results were obtained with 7^3 subdivisions, which confusion matrix is depicted in Fig. 20, with a test accuracy of 88,74%.

In [23] we can find the reported accuracy of the main relevant papers for both ModelNet10 and ModelNet40. In Table 1 we offer a summary with the performance of some recent or relevant works for this topic. It is important to note that many of these methods rely on deep learning and multi-view representations, contrary to the proposed method that is a global descriptor based on fractal properties of the objects.

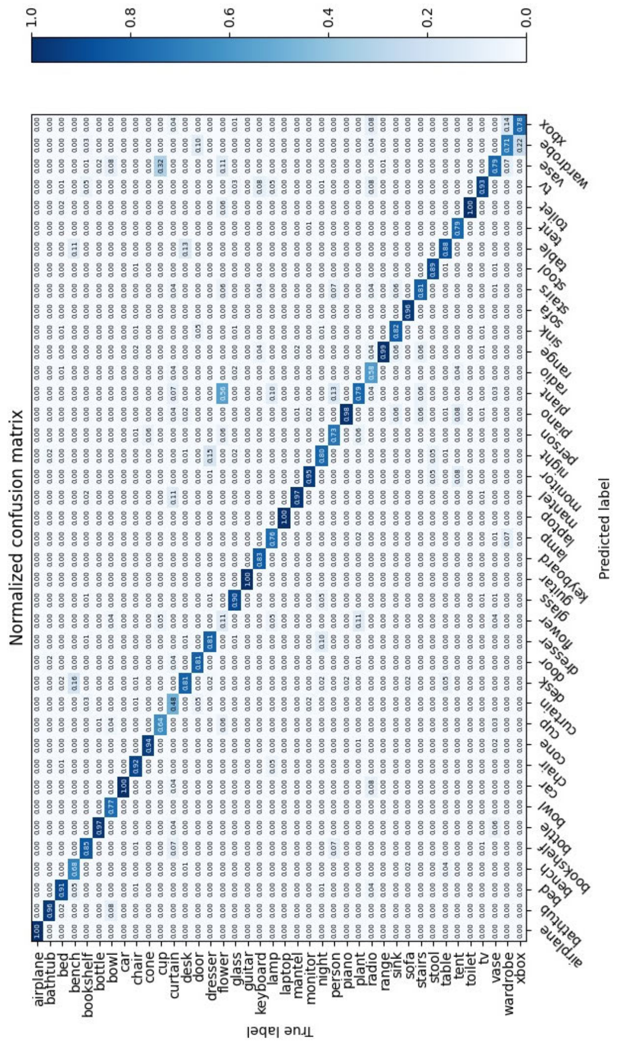


FIGURE 20. Confusion matrix for ModelNet40 with 7^3 subdivisions and SVM classifier. 88,74% accuracy.

TABLE 1. Comparative of accuracy results on ModelNet benchmark.

Paper	ModelNet10 Acc	ModelNet40 Acc
RotationNet [43]	98.46%	97.37%
SPNet [44]	97.25%	92.63%
SO-Net [18]	95,07%	93,40%
Point2Sequence [45]	95,30%	92.60%
Lonchanet [22]	94,37%	-
VFD	**92.84%**	**88.74%**
binVoxNetPlus [46]	92.32%	85.47%
Primitive-GAN [47]	92.20%	86.00%
VSL [48]	91.00%	84.50%
OrthographicNet [49]	88.56%	-
3DShapeNets [24]	83.50%	77.00%
PointNet [26]	77,6%	-

V. CONCLUSION AND FUTURE WORK

The feasibility of the fractal dimension as global descriptor to recognize 3D objects has been demonstrated through the experiments carried out. The results obtained for the datasets *ModelNet10* and *ModelNet40*, used as benchmarks, have been successful, with a test accuracy of 92,84% and 88,74% respectively, which make the VFD comparable to the deep learning-based methods of the state of the art.

One main advantage of our method is that it is inherently agnostic to the to the density of points of the sample, number of points in the input cloud, sensor of choice, and noise up to a level because it is based on the fractal dimension of a set of points. This feature allows VDF to provide competitive accuracy on classification tasks. Another advantage is that our method is able to work on real life, sensor-provided point clouds as it do not relies on intermediate, rendered representations. It also does not require any powerful, specific hardware to create the descriptor and perform classification tasks on a reasonable amount of time, unlike the deep learning-based approaches. Nonetheless, our descriptor has linear complexity as the size of the descriptor is directly related to the resolution of the voxel grid clustering, as there is one component on the descriptor for each voxel. The resolution is also linked to the accuracy, as more complex objects would require more resolution in order to capture the finest details.

Throughout this article, different methods for calculating fractal dimension have been mentioned. It seems logical that if the box counting method has succeed for 3D object recognition, other methods could work too and even improve the results or efficiency, so we plan to test different approaches as future work. In addition to that, we would explore method to limit the complexity of the descriptor in order to make it constant.

Another path to explore is to improve the implementation of the box counting method. We want to study the selection of a subset of points to approximate the line which slope is the fractal dimension, instead of applying least-mean squares minimization over all points. With this technique we could avoid the distortion produced by the leftmost points, as shown in Fig. 21.

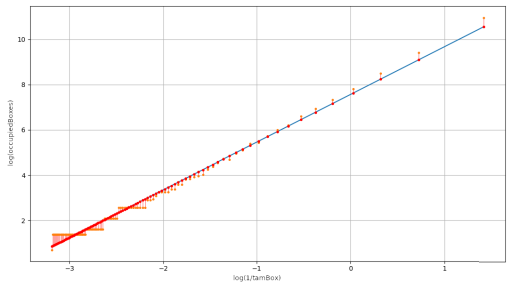


FIGURE 21. Errors with least-mean square minimization for box counting fractal dimension calculation.

ACKNOWLEDGMENT

Experiments were made possible by a generous hardware donation from NVIDIA.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[3] F. Luo, L. Zhang, B. Du, and L. Zhang, "Dimensionality reduction with enhanced hybrid-graph discriminant learning for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 8, pp. 5336–5353, Aug. 2020.

[4] K. Alhamzi, M. Elmogy, and S. Barakat, "3d object recognition based on local and global features using point cloud library," *Int. J. Adv. Comput. Technol.*, vol. 7, no. 3, p. 43, 2015.

[5] S. Xia, D. Chen, R. Wang, J. Li, and X. Zhang, "Geometric primitives in LiDAR point clouds: A review," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 685–707, 2020.

[6] N. Bayramoglu and A. A. Alatan, "Shape index SIFT: Range image recognition using local features," in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 352–355.

[7] A. Aldoma, F. Tombari, L. D. Stefano, and M. Vincze, "A global hypotheses verification method for 3d object recognition," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2012, pp. 511–524.

[8] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Dec. 2011, pp. 2987–2992.

[9] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Matching 3D models with shape distributions," in *Proc. Int. Conf. Shape Modeling Appl.*, 2001, pp. 154–166.

[10] C. Y. Ip, D. Lapadat, L. Sieger, and W. C. Regli, "Using shape distributions to compare solid models," in *Proc. 7th ACM Symp. Solid Modeling Appl. (SMA)*, 2002, pp. 273–280.

[11] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D recognition and pose using the viewpoint feature histogram," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 2155–2162.

[12] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 3212–3217.

[13] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski, "CAD-model recognition and 6DOF pose estimation using 3D cues," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCV Workshops)*, Nov. 2011, pp. 585–592.

[14] A. Aldoma, F. Tombari, R. B. Rusu, and M. Vincze, "OUR-CVFH–oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6DOF pose estimation," in *Proc. Joint DAGM German Assoc. Pattern Recognit. OAGM Symp.* Cham, Switzerland: Springer, 2012, pp. 113–122.

[15] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[16] C. R. Qi, O. Litany, K. He, and L. Guibas, "Deep Hough voting for 3D object detection in point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9277–9286.

[17] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz, "SPLATNet: Sparse lattice networks for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2530–2539.

[18] J. Li, B. M. Chen, and G. H. Lee, "SO-net: Self-organizing network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9397–9406.

[19] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel CNN for efficient 3D deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 963–973.

[20] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 945–953.

[21] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3813–3822.

[22] F. Gomez-Donoso, A. Garcia-Garcia, J. Garcia-Rodriguez, S. Orts-Escolano, and M. Cazorla, "LonchaNet: A sliced-based CNN architecture for real-time 3D object recognition," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 412–418.

[23] Princeton. *Princeton Modelnet*. Accessed: Apr. 14, 2019. [Online]. Available: http://modelnet.cs.princeton.edu/

[24] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.

[25] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 922–928.

[26] A. Garcia-Garcia, F. Gomez-Donoso, J. Garcia-Rodriguez, S. Orts-Escolano, M. Cazorla, and J. Azorin-Lopez, "PointNet: A 3D convolutional neural network for real-time object class recognition," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 1578–1584.

[27] F. Gomez-Donoso, F. Escalona, and M. Cazorla, "Par3DNet: Using 3DCNNs for object recognition on tridimensional partial views," *Appl. Sci.*, vol. 10, no. 10, p. 3409, May 2020.

[28] T. Le and Y. Duan, "PointGrid: A deep network for 3D shape understanding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9204–9214.

[29] 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. 1–11, Jul. 2017.

[30] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2088–2096.

[31] C. R. Qi, H. Su, M. NieBner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5648–5656.

[32] A. Sharma, O. Grau, and M. Fritz, "VConv-DAE: Deep volumetric shape learning without object labels," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 236–250.

[33] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," 2016, *arXiv:1608.04236*. [Online]. Available: http://arxiv.org/abs/1608.04236

[34] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 82–90.

[35] O. M. Bruno, R. de Oliveira Plotze, M. Falvo, and M. de Castro, "Fractal dimension applied to plant identification," *Inf. Sci.*, vol. 178, no. 12, pp. 2722–2733, Jun. 2008.

[36] G. Shen, "Fractal dimension and fractal growth of urbanized areas," *Int. J. Geographical Inf. Sci.*, vol. 16, no. 5, pp. 419–437, Jul. 2002.

[37] C. Gómez, Á. Mediavilla, R. Hornero, D. Abásolo, and A. Fernández, "Use of the Higuchi's fractal dimension for the analysis of MEG recordings from Alzheimer's disease patients," *Med. Eng. Phys.*, vol. 31, no. 3, pp. 306–313, Apr. 2009.

[38] T. Higuchi, "Approach to an irregular time series on the basis of the fractal theory," *Phys. D, Nonlinear Phenomena*, vol. 31, no. 2, pp. 277–283, Jun. 1988.

[39] B. Mandelbrot, "How long is the coast of Britain? Statistical self-similarity and fractional dimension," *Science*, vol. 156, no. 3775, pp. 636–638, 1967.

[40] I. M. Rian and M. Sassone, "Fractal-based generative design of structural trusses using iterated function system," *Int. J. Space Struct.*, vol. 29, no. 4, pp. 181–203, Dec. 2014.

[41] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," 2018, *arXiv:1801.09847*. [Online]. Available: http://arxiv.org/abs/1801.09847

[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[43] A. Kanezaki, Y. Matsushita, and Y. Nishida, "RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5010–5019.

[44] M. Yavartanoo, E. Y. Kim, and K. M. Lee, "Spnet: Deep 3D object classification and retrieval using stereographic projection," in *Proc. Asian Conf. Comput. Vis.* Cham, Switzerland: Springer, 2018, pp. 691–706.

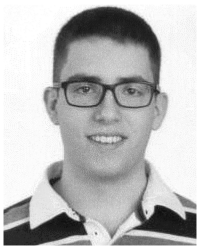
[45] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, "Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 8778–8785.

[46] C. Ma, W. An, Y. Lei, and Y. Guo, "BV-CNNs: Binary volumetric convolutional networks for 3D object recognition," in *Proc. Brit. Mach. Vis. Conf.*, 2017, p. 4.

[47] S. H. Khan, Y. Guo, M. Hayat, and N. Barnes, "Unsupervised primitive discovery for improved 3D generative modeling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9739–9748.

[48] S. Liu, L. Giles, and A. Ororbia, "Learning a hierarchical latent-variable model of 3D shapes," in *Proc. Int. Conf. 3D Vis. (3DV)*, Sep. 2018, pp. 542–551.

[49] H. Kasaei, "OrthographicNet: A deep learning approach for 3D object recognition in open-ended domains," 2019, *arXiv:1902.03057*. [Online]. Available: http://arxiv.org/abs/1902.03057



JOSE FRANCISCO DOMENECH received the Computer Science degree from the University of Alicante, Spain, in 2020.

He has participated in this research as his graduation project. He also studied with the Czech Technical University of Prague, Czech Republic, in 2018, as an Erasmus Student. At the professional level, he has been working, from 2019 to 2020 in web and database development projects, focused in market studies. His research interests include 3D object recognition, segmentation and scene understanding and domestic and social robotics.

FELIX ESCALONA was born in Alicante, Spain, in 1994. He received the B.S. degree in computer science, earning the extraordinary end-of-degree prize, and the M.S. degree in robotics from the University of Alicante, in 2016 and 2017, respectively. He is currently pursuing the Ph.D. degree in computer science with the University of Alicante, Spain, with an FPU scholarship.

His research interests include 3D object recognition, segmentation and scene understanding, and domestic and social robotics.

FRANCISCO GOMEZ-DONOSO received the B.S. degree in computer science from the University of Alicante, Spain, in 2014, and the master's degree in robotics and automatic, in 2015. He is currently pursuing the Ph.D. degree in computer science program with the University of Alicante. Regarding his experience as a Scientist, he has published more than 35 articles in high-impact journals and conferences. His main research interests include human–computer interaction, deep learning and machine learning, and tridimensional data processing.

MIGUEL CAZORLA (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the University of Alicante, in 2000.

He was a Computer Engineer with the University of Alicante, in 1995. In 1995, he joined as an Assistant Professor with the University of Alicante, where he has been a Full Professor, since 2017. He has completed several stays at foreign institutions (Carnegie Mellon University, the University of Sydney, and the University of Edinburgh). He has published more than 50 articles indexed in JCR (with more than 20 in Q1) and more than 100 publications in national and international conferences. He has supervised 11 Ph.D. theses, where he is a Principal Investigator in several national projects (CICYT, Challenges), and has made multiple transfer contracts with the industry.

Dr. Cazorla is a member of different program committees of national and international conferences. His research line has always been centered on computer vision. From the beginning, he applied these skills to try to solve robotic tasks. Almost since its inception in research has worked in the processing of 3D data. In recent years, he has diversified his lines to apply deep learning techniques to different areas (medical image, object recognition, depth estimation, identification of traffic objects, and so on.). All his research in recent years, he has been focused on social robotics: application of all these techniques to help dependents.