Learning with Latent Representations of 3D Data: from Classical Methods to 3D Deep Learning

Chengzhi Wu

Vision and Fusion Laboratory
Institute for Anthropomatics
Karlsruhe Institute of Technology (KIT), Germany
chengzhi.wu@kit.edu

Technical Report IES-2019-11

Abstract

3D data contain rich information about the full geometry of objects or scenes. Learning tasks on them have always been considered as hard ones in the computer vision community due to their extreme high dimensionality. Hence, latent representations of 3D geometries are often used to lower the data dimensionality for better parameterization and easier computation. In this report, we make a brief review on those latent representations obtained via different methods including classical ones and the emerging neural learning-based ones. Furthermore, the nowadays widely used deep learning methods have also been more closely investigated regarding their applications on various 3D data formats. The possibility of combing those two kinds of methods has also been addressed.

1 Introduction

3D data analysis has always been an interesting yet challenging research topic for computer vision researchers. Learning latent information from them is vital

to lots of advanced technology applications including robotics, autonomous driving, virtual reality and augmented reality. Lots of classical methods have been proposed to extract latent representations from 3D data. Those latent representations can be images, graphs, histograms, or even vectors [4]. Classical methods usually focus more on generating latent representations of 3D shapes. Those generated latent representations are sometimes also referred as shape descriptors.

In recent years, neural networks have been proved to be one of the most powerful learning algorithms for computer vision tasks, especially on 2D Euclidean data. Implicitly learned feature maps or bottleneck feature vectors have been used for classification, detection, or segmentation tasks. Later on, similar methods have been proposed on 3D Euclidean data with minor adaptions. However, those learning algorithms cannot be straightforwardly extended to Non-Euclidean data due to their non-grid data structure. Different special neural network architectures for 3D Non-Euclidean data therefore have been more meticulously designed and proposed, while input, output, latent representations, or even network operations have been more artfully defined.

This report is structured as follows. In Section 2, we briefly review the most common 3D data formats. Latent representations learned by classical methods or neural learning-based methods are reviewed in Section 3. Section 4 gives a more detailed review on the application of deep learning a) for ML tasks on 3D data and b) for the generation of latent representations that can be used by different methods later on. Conclusion and future outlook are presented in Section 5.

2 Overview of 3D data format

3D data have lots of different formats depending on its source. They are usually categorized into 2 subsets, Euclidean data, which mainly include multi-view images, RGB-D images, volumetric voxels or octrees; and Non-Euclidean data, which mainly include point clouds and meshes. Euclidean data are usually of rasterized forms, they have regular grids. For example, images are composed of pixels which are well aligned and always have same number of neighbours.

Non-Euclidean data are usually more of geometric forms, they do not have regular grids. For example, with geometric metrics, the distance between two vertices on a mesh should be computed as their geodesic distance on the manifold, other than the direct Euclidean distance. In this section, different 3D data formats are briefly reviewed and compared.

2.1 Euclidean data

Multi-view images: 3D data may be presented as a combination of multiple 2D images captured for the 3D object from different view points [38]. Learning with this format, the noise effect from incompleteness, occlusion and illumination problems can be well reduced. All the input views jointly optimize the functions to represent the whole 3D shape. However, this format requires too many input sources and is usually too expensive for industrial use. The question of how many views are sufficient to represent a shape is also still open.

RGB-D images: With the development of RGB-D sensors, e.g., Microsoft Kinect, more and more industrial applications are using RGB-D images as the input data format for their tasks. This data format provides an additional depth map along with the normal 2D RGB color information. Comparing to other 3D data formats, there are more RGB-D data format available due to its inexpensiveness [7].

Volumetric data: Same as 2D shapes can be rasterized into pixels, 3D shapes can also be rasterized into voxels. In this case, 3D shapes are encoded by those occupied voxels. Despite the simplicity of the voxel-based representation, it suffers from keeping the intrinsic properties of 3D shapes and the smoothness of their surfaces [34]. It also requires high memory storage and has high computation complexity, which makes volumetric format not appropriate for high-resolution data.

2.2 Non-Euclidean data

Point clouds: A point cloud is a set of unstructured points that approximate the geometry of an object. However, if we only consider the local structure of

the object, those subsets may also be considered as Euclidean since they have a global parameterization and are usually represented by a normal system of coordinates. It depends on the metrics method that is used. But most tasks still focus on the global structure for shape recognition, matching or retrieval, hence point clouds are still classified as Non-Euclidean data format in most cases. Nowadays we have multiple choices of 3D sensors to generate point clouds, e.g., Ensenso or Zivid, they usually do single-shot and capture the whole scene. Therefore, different from other formats, preprocessing steps such as noise filtering or scene segmentation are usually required for point clouds of 3D shapes.

Meshes/graphs: A polygon mesh is a collection of vertices, edges and faces that defines the shape of a polyhedral object in 3D computer graphics and solid modeling. With an appropriate number of vertices, meshes can give extremely accurate geometric information of 3D shapes. The vertices in a mesh have certain connectivities, which makes mesh a special case of graph. The process of generating an approximate watertight mesh from a random connected graph is called 3D shape completition or inpainting. Although meshes contains rich information of 3D shapes, it is really a challenging task to learn on them directly due to its irregularity. In most relevant researches, the spectral properties of the graphs and meshes are utilized to learn latent features after applying a graph Laplacian eigen-decomposition.

Continuous space function: Continuous space functions are a very special data format. It uses a mathematical function to represent the 3D shape directly and precisely. It is also referred as level set or signed distance function (SDF) with minor definition modification. Input a coordinate in the defined space, a SDF outputs a value whose sign (positive or negative) denotes that this point is outside or inside the shape boundary. For example, if the output space of a SDF is defined between $[-1, 1]$, the whole function may be considered as a mapping function $f : \mathbb{R}^3 \rightarrow [-1, 1]$. If 0 is defined as the cutoff boundary, then all the points whose coordinates yield an output between $[-1, 0]$ after the mapping means they are inside the object surface, and vice versa. However, only simple shapes like cube, heart, donuts or lemon can be easily denoted with a SDF. It is more often impossible to find such a function for a slightly complex shape. Thus this data format is less explored comparing to others.

Table 2.1: Property comparison of different 3D data formats

		Accuracy	Affability to NNs		Geometric manipulability	
			input/output	computation	deform etc.	constrains
Euclidean	image-based	controversial	great	great	poor	poor
	voxel-based	poor	good	poor	controversial	poor
Non-Euclidean	point clouds	good	good	controversial	good	good
	meshes/graphs	great	poor	controversial	great	good
	continuous space function	great	poor	poor	controversial	poor

2.3 Property comparison

It is impossible to say which data format is the best 3D data format. Apart from the accuracy requirements, to better make use of the 3D information, it is usually excepted that the data should be geometrically manipulable (deformation, interpolation, etc.) and convenient to impose structural constraints. On the other hand, since we are interested in applying deep learning algorithms on them, the data should also be able to be easily formulated as the input/output to neural networks and make fast forward/backward propagation computation possible. Here, based on the state-of-the-art researches, we summarize the overall rating subjectively on these properties of different 3D data formats in Table 2.1. In most cases, people will just use the most appropriate data format for their tasks according to the input source limitation, computation ability, and accuracy and robustness requirements.

3 Latent representations of 3D data

The process of acquiring latent representations from input data is essentially a mapping process. It maps the input data from its original data space to another latent space, which are usually lower dimensional. In statistics definition, latent representations (or, latent variables) are variables that are not directly observed but are rather inferred through a mathematical model from other variables that are directly observed and measured. Although multi-view images or volumetric data may be regarded as a special mapping method that maps the original geometric data into a lower dimensional space, those data representations are usually not considered as latent ones since we can still observe shape properties directly on them. Hence, in this report, we regard them as other kinds of data formats and not as latent representations.

Before the recent upsurge of deep learning, there were already many other classical mathematical methods that try to encode 3D data, mostly on 3D shapes. For 3D shapes, the latent representations of them are also called as shape descriptors. In this section, we first make a brief overview on those classical

methods and the shape descriptors they generated, then probe into the latent representations learned with neural networks.

3.1 Classical methods

Classical methods usually have very strong mathematical background, involving strict mathematical formulas and deductions. Therefore the encoding results from them are usually deterministic. There are numerous classical methods that try to learn latent representations from 3D data, whether on Euclidean formats or Non-Euclidean formats. Here we just summarize and list some most known ones that may be related or helpful to our future work.

Ray-based sampling with spherical harmonics: In order to characterize shapes of functions on a sphere by just a few parameters, spherical harmonics [9] were proposed as a suitable tool. The magnitudes of complex coefficients, which are obtained by applying the fast Fourier transform on the sphere to the samples, are regarded as vector components. Thus, the ray-based feature vector is represented in the spectral domain, where each vector component is formed by taking into account all original input.

Laplacian spectral eigenvectors: In addition to considering the connectivity of nodes and edges in a graph, mesh Laplacian operators take into account the geometry of a surface (e.g. the angles at the nodes). For a manifold triangle mesh, the Laplace-Beltrami operator is used to represent the intrinsic geometric structure. After applying the Laplacian eigen-decomposition, the original shape may be represented by its spectral eigenvectors, which makes mesh processing [24] and surface editing [25]possible.

Heat kernel signature: A heat kernel signature (HKS) is a shape descriptor obtained via spectral shape analysis methods and in use for deformable shape analysis. It is based on heat kernel, which is a fundamental solution to the heat equation [27]. For each point in the shape, HKS defines its feature vector representing the point's local and global geometric properties. HKS is one of the many recently introduced shape descriptors which are based on the LaplaceBeltrami operator associated with the shape. There are other relevant

shape descriptors including global point signature (GPS), biharmonic signature (BS), wave kernel signature (WKS).

Skeleton-based 3D descriptor: Skeletons derived from solid objects can be regarded as intuitive object descriptions. They are able to capture the most important information about the shape structure. Sundar et al. [28] presented a framework for skeletonization and 3D object retrieval. Skeleton-based 3D descriptor is widely used in animation and film industrial nowadays due to its ideal parameterized control on the shape joints.

Primitive-based CAD model descriptor: 3D shapes may be approximately assembled by composing simple volumetric primitives including cuboids, cylinders and spheres. The shapes from one category usually have similar primitive representations. Using this abstract representation, interpolation between the obtained latent representations may provide a consistent parsing across shapes in one certain category.

3.2 Neural learning-based methods

Comparing to the classical methods, neural learning-based methods are less deterministic since they have more stochastic calculations involved. The final parameters of a trained neural network may be slightly different even though all the settings are identical in multiple trainings.

Actually, the latent representations learned via neural networks are seldom of particular concern in most computer vision tasks, while they have always been implicitly used. A good example would be the bottleneck features in transfer learning. In transfer learning, we take a pre-trained model including network and weights, then remove the last few fully connected (FC) network and construct our own in place of it. When the training starts on the new data set, usually the original network parameters before the FC network are frozen and only the newly added FC network are trained. Here the input to the FC network is referred as bottleneck features. They represent the latent features learned from the last convolution layer in the network. Surely we can take the feature maps from any previous layer and name them as bottleneck features or latent representations, but in most cases we are more interested in a vector representation, thus a flattened

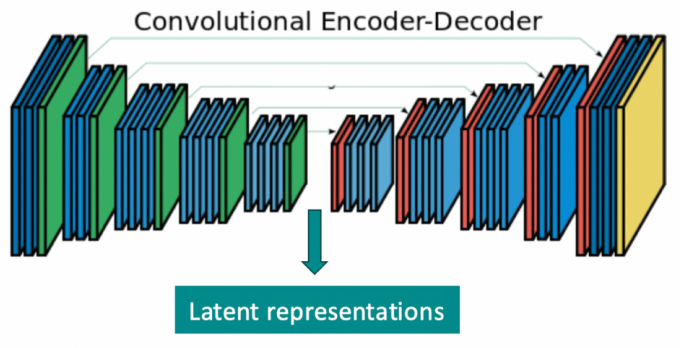

Figure 3.1: The basic structure of a neural encoder-decoder. The feature maps/vectors learned inside the network may be regarded as latent representations.

vector bottleneck feature are more often taken and used. But, still, the properties of bottleneck features themselves are really less explored.

Also in generation tasks, latent representations are also crucial to learning. A typical generative adversarial network (GAN) may take a vector from the latent space as the input to generate pseudo real world data. Interpolating between the input latent vectors, a continuous reshaping or deforming output can usually be observed.

Figure 3.1 gives a brief idea how latent representations are learned within a neural encoder-decoder. A more detailed survey of how latent representations of 3D shapes are obtained and utilized with deep learning methods is given in the next section.

4 Deep learning on 3D data

4.1 Learning on 3D Euclidean data

In order to duplicate the success of deep learning techniques from the 2D domain to the 3D domain, it is easy to see that we can use 3D Euclidean data directly

for learning purposes. In case of only 3D Non-Euclidean data are provided, we can always convert them into Euclidean formats with a certain information loss. Due to its simplicity and convenience, this converting process has been widely utilized to create rasterized data to fit in the Euclidean neural network architectures ever since the emerging of deep learning, even till now.

4.1.1 Image-based representations

When using RGB-D images or multi-view images as the input for deep learning tasks, it is often required to have multiple input channels or even multiple CNN streams to process the data. For example, [5] used a two-stream CNNs on RGB-D data for 3D object recognition tasks. The learned latent features from two streams were fused together in one later FC layer and the classification result was given after a further softmax layer. A more interesting method was proposed in [2], in which the idea of transfer learning was combined with the method used in [5]. It used four separate CNNs to train the four channels in the RGB-D data, while the weights were transferred from each network to another. Their results indicated that the depth information carries valuable information about shapes.

More processing streams will be needed for the multi-view images data format. MVCNN [26] processed rendered 12 views of a 3D object separately. Then a max pooling operation was applied in the view-pooling layer to get a compact latent representation for the whole shape. In [37], a multi-branch CNN has been designed to use rendered depth maps from different views of the object as input. Each branch returned a feature vector that contributes to the final classification. Apart from single value output recognition/classification tasks, this format has also been used for other more complex tasks. Kalogerakis et al. [11] designed a neural network for segmenting 3D objects into their labeled semantic parts by learning from their multiple 2D projections. Local shape descriptors from part correspondences have also been learned with a multi-view convolutional network [10]. Even 3D shape reconstruction via multi-view convolutional networks has also been studied from sketches in [13].

4.1.2 Volumetric data

Regular 2D convolution operations have been naturally extended to 3D convolution operations by applying 4D convolutional kernels, certain network architectures have also been proposed. VoxNet [15] first converted the point clouds of shapes into voxels according to their occupancy in the space. Then this volumetric data was used as input to their neural network for shape classification. A similar method has been propose in 3DShapeNets [33] except they got the volumetric data from depth maps. As a followed work, Seaghat et al. [23] modified the architecture of VoxNet by incorporating the orientation of 3D objects in the learning process.

Regarding the synthesis tasks with 3D volumetric data, in [32], by extending the idea of GAN in the 2D domain, volumetric generative adversarial networks have also been proposed. In McRecon network structure [8], foreground masks have been used as weak supervision through a raytrace pooling layer for 3D reconstruction. There are also octree-based methods which only consider the occupied grids in a more memory efficient way including OctNet[22] and O-CNN [29].

4.2 Learning directly on 3D Non-Euclidean data

As mentioned in the last subsection, people can always convert 3D Non-Euclidean data to Euclidean formats for convenient neural network architecture designs since the technical maturity of similar methods in 2D domain are already quite high. However, object information will be inevitably lost during the converting process. The best way to prevent this information loss is learning directly on 3D Non-Euclidean data, in which special ways to define the input, output, or even the operations used in the networks are usually required.

4.2.1 Point clouds

The very first proposed deep learning-based method of directly using 3D point clouds data for shape analysis tasks is PointNet [20]. It used (x, y, z) coordinates of points as input to the network, then an additional spatial transform network

was performed as a pre-processing step. After that, lots of weights-sharing fully connected layers were added to compute point-wise features. Finally, a max-pooling layer was used to aggregate the global information and output a 1024 dimensional latent feature vector for classification tasks. For segmentation tasks, the global shape feature and the point-wise features were concatenated for predicting point-wise segmentation result. Despite the competitive results achieved by PointNet, it still failed to take full advantage of the local features in point clouds. Their subsequent work PointNet++ [21] tried to address this point by grouping the points with different scales, performing PointNet on them separately in order to aggregate different scale features. To better aggregate the information in the real local area, aggregate operations similar to the convolution operations have also been proposed, such as EdgeConv defined in [31] or X-Conv defined in [12]. Both of them took a certain number of neighbours of each point into consideration and performed the aggregating operation point-wise. With this operations, the learned final latent representation also contains local information implicitly.

In 3D point clouds synthesis field, [1] proposed a deep auto encoder (AE) with high reconstruction quality and generalization. Generative adversarial networks (GANs) and Gaussian Mixture Models (GMMs) have also been trained in the latent space of their AEs respectively. Similarly, FoldingNet [35] proposed a point clouds auto-encoder via deep grid deformation with graph-based encoders, in which special perceptron layers were defined as folding operations. Regarding the upsampling task for sparse point clouds, PU-Net was especially designed with convolution operations defined in the latent feature space [36].

4.2.2 Meshes

At first glance, triangular meshes give people the illusion that 2D convolutional kernels may be directly applied. However, these rasterized kernels are only applicable to Euclidean data due to their structure shift invariance property. In order to perform convolution locally, appropriate local patches need to be defined. Geodesic CNN (GCNN) [14] constructed local patches in local polar coordinates to ensure their structure non-position-dependent. Values of the functions around each vertex in the mesh are mapped into local polar coordinates

using the patch operator, thus geodesic convolution may be applied on those patches. Later on, Anisotropic CNN (ACNN) [3] was proposed to tackle the limitations in GCNN. It constructed a simpler pattern of local patches, which are independent to the injectivity radius of meshes. Rather than using a fixed kernel pattern as in GCNN and ACNN, MoNet [18] were proposed to define a vertex-wise locally weighted coordinate system, on which parametric kernels were applied to define the weighting functions. With this definition, GCNN and ACNN may be considered as special cases of MoNet with certain constraints.

Except for those methods defined on the spatial domain, methods defined on the spectral domain have also been proposed. For example, [6] first computed heat kernel descriptors of shapes based on their heat kernel signatures (HKS), then the descriptors were fed into two neural networks with target value using Eigen-shape Descriptor and Fisher-shape Descriptor, respectively. The final deep shape descriptor is formed by concatenating nodes in hidden layers. [30] proposed a similar pipeline with local point signature (LPS) features. Multi-scaled vertex spectral images were generated by packing the 16-dimensional LPS in a compact manner, and then fed into a CNN to generate the final shape descriptor. Those methods show the possibility that shape properties obtained via classical methods may be further utilized with the deep learning methods to get a better latent representation, with which better performance of different tasks may be achieved.

4.2.3 Continuous space function

Continuous space function (CSF) or signed distance function (SDF) is a really less explored data format. Although it provides high accuracy, it is usually impossible to easily find a function that matches a slightly complex object. Fortunately, neural networks are "universal approximators" and can mimic any continuous function to the degree that the network size permits.

Early this year, DeepSDF [19] was proposed to learn a continuous SDF representation for a 3D shape, which encoded a shape's boundary as the zero-level-set of the learned function that explicitly divided the space into shape interior and shape exterior. Deep Level Sets [17] also deployed a similar idea to represent the output as an oriented level set of a continuous embedding function with the

help of deep neural networks. In a more recent paper, Mescheder et al. [16] proposed Occupancy Networks, which also used a network to mimic functions that define the shape boundaries. An interesting adaption in their method is that rather than a signed value, the output of the network is a real value between 0 and 1, which indicates the occupancy possibility of a certain point in that space position. Although all those methods usually need a post-processing step to visualize the shapes, the reconstruction performance of them are usually qualitatively better than the performance of classical methods that only work for point clouds or meshes.

5 Conclusion

In this report, we first briefly review the most used 3D data formats, including both the Euclidean ones and the Non-Euclidean ones. Secondly, latent representations or shape descriptors obtained via classical methods and deep neural networks have been reviewed and discussed. While several classical methods have been addressed, more efforts have been put into investigating the neural learning-based methods. Latent representations of different 3D data formats learned with various network architectures have been reviewed and discussed, the possibility of combing classical methods and neural learning-based methods has also been especially addressed. Although within the deep learning scope, the dominant approaches that utilized for various computer vision tasks nowadays are still usually based on images or other Euclidean data, we hope that with a better learning and understanding of the latent representations of 3D shapes, more efficient architectures may be proposed and better performance may be achieved with them in the future.

References

[1] P. Achlioptas et al. “Learning Representations and Generative Models for 3D Point Clouds”. In: *International Conference on Machine Learning (ICML)* (2018).

[2] L. Alexandre. “3D Object Recognition Using Convolutional Neural Networks with Transfer Learning between Input Channels”. In: *Intelligent Autonomous Systems* 13 (2016), pp. 889–898.

[3] D. Boscaini et al. “Learning Shape Correspondence with Anisotropic Convolutional Neural Networks”. In: *NIPS* (2016).

[4] B. Bustos and D. Keim and D. Saupe, T. Schreck, and D. Vrani. “Feature-based Similarity Search in 3D Object Databases”. In: *ACM Computing Surveys (CSUR)* 37 (2005), pp. 345–387.

[5] A. Eitel et al. “Multimodal Deep Learning for Robust RGB-D Object Recognition”. In: *Intelligent Robots and Systems (IROS)* (2015), pp. 681–687.

[6] Y. Fang et al. “3D Deep Shape Descriptor”. In: *CVPR* (2015), pp. 2319–2328.

[7] M. Firman. “RGBD Datasets: Past, Present and Future”. In: *CVPR Workshop on Large Scale 3D Data: Acquisition, Modelling and Analysis* (2016).

[8] J. Gwak et al. “Weakly supervised 3D Reconstruction with Adversarial Constraint”. In: *International Conference on 3D Vision (3DV)* (2017).

[9] D. Healy et al. “FFTs for the 2-sphere Improvements and Variations”. In: *Journal of Fourier Analysis and Applications* 4 (2003), pp. 341–385.

[10] H. Huang et al. “Learning Local Shape Descriptors from Part Correspondences With Multi-view Convolutional Networks”. In: *CVPR* (2017).

[11] E. Kalogerakis et al. “3D Shape Segmentation with Projective Convolutional Networks”. In: *CVPR* (2017).

[12] Y. Li et al. “PointCNN”. In: *arXiv preprint arXiv:1801.07791* (2018).

[13] Z. Lun et al. “3D Shape Reconstruction from Sketches via Multi-view Convolutional Networks”. In: *Proceedings of the International Conference on 3D Vision (3DV)* (2017).

[14] J. Masci et al. “Geodesic Convolutional Neural Networks on Riemannian Manifolds”. In: *IEEE International Conference on Computer Vision Workshop (ICCVW)* (2015).

[15] D. Maturana and S. Scherer. “VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition”. In: *International Conference on Intelligent Robots (IROS)* (2015), pp. 922–928.

[16] L. Mescheder et al. “Occupancy Networks: Learning 3D Reconstruction in Function Space”. In: 2019.

[17] M. Michalkiewicz et al. “Deep Level Sets: Implicit Surface Representations for 3D Shape Inference”. In: *ArXiv* abs/1901.06802 (2019).

[18] F. Monti et al. “Geometric Deep Learning on Graphs and Manifolds using Mixture Model CNNs”. In: *CVPR* (2017).

[19] J. Park et al. “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”. In: *CVPR* (2019).

[20] C. Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *CVPR* (2017).

[21] C. Qi et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *NIPS* (2017).

[22] G. Riegler, A. Ulusoy, and A. Geiger. “OctNet: Learning Deep 3D Representations at High Resolutions”. In: *CVPR* (2017).

[23] N. Sedaghat et al. “Orientation-boosted voxel nets for 3D object recognition”. In: *British Machine Vision Conference (BMVC)* (2017).

[24] O. Sorkine. “Laplacian Mesh Processing”. In: *Eurographics 2005 - State of the Art Reports* (2005).

[25] O. Sorkine et al. “Laplacian Surface Editing”. In: *Symposium on Geometry Processing* (2004).

[26] H. Su et al. “Multi-view Convolutional Neural Networks for 3D Shape Recognition”. In: *ICCV* (2015), pp. 945–953.

[27] J. Sun, M. Ovsjanikov, and L. Guibas. “A Concise and Provably Informative Multi-Scale Signature-Based on Heat Diffusion”. In: *Computer Graphics Forum* 28 (2009), pp. 1383–1392.

[28] H. Sundar et al. “Skeleton-based Shape Matching and Retrieval”. In: *Proceedings of the Shape Modeling International* (2003), pp. 45–53.

[29] P. Wang et al. “O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis”. In: *SIGGRAPH* (2017).

[30] Y. Wang et al. “A Robust Local Spectral Descriptor for Matching Non-Rigid Shapes with Incompatible Shape Structures”. In: *CVPR*. 2019.

[31] Y. Wang et al. “Dynamic Graph CNN for Learning on Point Clouds”. In: *arXiv preprint arXiv:1801.07829* (2018).

[32] J. Wu et al. “Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling”. In: *NIPS* (2016).

[33] Z. Wu et al. “3D ShapeNets: A Deep Representation for Volumetric Shapes”. In: *CVPR* (2015).

[34] Y. Xiang et al. “Data-driven 3D Voxel Patterns for object category recognition”. In: *CVPR* (2015), pp. 1903–1911.

[35] Y. Yang et al. “FoldingNet: Point Cloud Auto-encoder via Deep Grid Deformation”. In: *CVPR* (2018).

[36] L. Yu et al. “PU-Net: Point Cloud Upsampling Network”. In: *CVPR* (2018).

[37] P. Zanuttigh and L. Minto. “Deep learning for 3D shape classification from multiple depth maps”. In: *International Conference on Image Processing (ICIP)* 155 (2017), pp. 3615–3619.

[38] S. Zhi et al. “Toward Real-time 3D Object Recognition: A Lightweight Volumetric CNN Framework Using Multitask Learning”. In: *Computers and Graphics* 71 (2017), pp. 199–207.