IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Addressing the Sim2Real Gap in Robotic 3D Object Classification

Jean-Baptiste Weibel, Timothy Patten and Markus Vincze

***Abstract*— Object classification with 3D data is an essential component of any scene understanding method. It has gained significant interest in a variety of communities, most notably in robotics and computer graphics. While the advent of deep learning has progressed the field of 3D object classification, most work using this data type are solely evaluated on CAD model datasets. Consequently, current work does not address the discrepancies existing between real and artificial data. In this work, we examine this gap in an indoor service robotic context by specifically addressing the problem of classification when transferring from artificial CAD models to real reconstructed objects. This is performed by training on ModelNet (CAD models) and evaluating on ScanNet (objects extracted from reconstructed rooms). We show that standard methods do not perform well in this task. We thus introduce a method that carefully samples object parts that are reproducible under various transformations and hence robust. Using graph convolution to classify the composed graph of parts, our method improves upon the baseline. Code is publicly available at https://rgit.acin.tuwien.ac.at/jb.weibel/cad2real_object_clf**

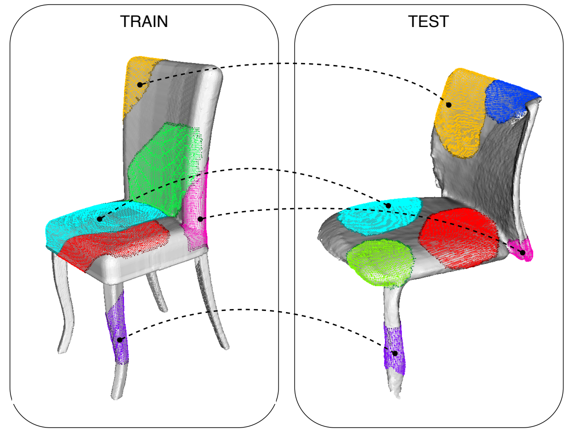

Fig. 1. Creating reproducible object parts with similar representations on all sources of data enables better transfer from artificial to real objects.

I. Introduction

Whether to recommend the most suitable CAD model to a designer or to enable a service robot to decide where to place objects when tidying a room, 3D object classification is an essential task. Research in this area has greatly benefited from the wide availability of 3D CAD models as well as the accessibility of depth sensors, as this has established a large amount of data to apply geometric reasoning.

Deep learning has profoundly transformed computer vision in recent years, and in particular, object classification has seen spectacular improvements. There has been steady interest for applying these methods for 3D data but introducing geometric reasoning in deep learning is not without its pitfalls. Typical deep learning approaches cannot handle rotated objects and real-world objects might be observed in arbitrary poses. Some methods use the statistical distribution of the data to transform the unknown object to a canonical pose for the deep network [10], [18]. However, inaccessible viewpoints, partial occlusions, supporting surfaces, and over- or under-segmentation observed in real-world data all contribute to modifying the statistical distribution of data that these methods expect, thus hindering their performance. This problem is particularly common for data obtained by indoor service robots. Most deep networks also expect a fixed size input. This is achieved by rescaling, however, applying this to an occluded object can lead to a significant difference in the final fixed size representation. Consider how rescaling and centering a model airplane to the unit sphere and the same model with one wing missing would produce vastly different coordinates. The effect becomes prevalent when transferring from CAD models to real-world objects, as scale information is not available during training since most CAD models are scaleless.

In this work, we develop a method for 3D object classification based on object parts that are reproducible under orientation or scale changes and can be defined for any level of occlusion, as shown in Figure 1. It should be noted that we define parts as a *continuous subset* of the original object without any specific semantic meaning. Indeed, semantic parts such as a cup handle or a chair leg are also likely to be occluded and impossible to recover from the original objects, whereas our non-semantic parts can always be defined. Once parts are extracted, a rotation-invariant representation is computed through the use of a reproducible local reference frame. Finally, a graph-convolution based architecture is used to classify the graph of parts. This approach is chosen because of its ability to incorporate information about the neighborhood of the parts without needing to define a global orientation or to know about the object delineation.

In summary, our contributions are the introduction of:

1) a carefully designed angle-based sampling procedure that creates object parts reproducible under various rotation, scale and occlusion and,
2) a general graph-based learning architecture for classification that preserves the relevant properties of 3D object parts.

These aspects allow us to achieve high performance when

The research leading to these results has received funding from the Austrian Science Foundation (FWF) under grant agreement No. I3968-N30 HEAP and No. I3969-N30 InDex

All authors are with the Vision for Robotics Laboratory, Automation and Control Institute, TU Wien, 1040 Vienna, Austria. {weibel, patten, vincze}@acin.tuwien.ac.at

transferring from artificial to real-world data. In particular, our approach transfers from the ModelNet dataset [33] to objects segmented from the reconstructed scenes in the ScanNet dataset [6] better than previous point cloud-based methods. Our approach also outperforms the baseline 3D method PointNet [18] when training and testing on ScanNet, which further demonstrates the value of careful part design and the inclusion of geometric priors.

The remainder of the paper is organized as follows. Section II reviews the relevant literature. Section III describes our approach for creating a graph of parts and subsequent learning for object classification. Section IV presents results of our method in comparison to existing methods, both on artificial data, real data, and when transferring from the former to the latter. Finally, Section V concludes and discusses future work.

II. Related Work

This section will first discuss approaches for geometric deep learning, and then approaches specifically designed for noisy data.

A. Geometric Deep Learning

Due to the way cameras perceive the world, 3D object classification is mostly concerned with surfaces (i.e. a 2D manifold in 3D space) rather than volumes. These surfaces are best represented with meshes, which is a specific type of graph. The branch of deep learning concerned with such data has recently received much attention from fields such as computer graphics as it focuses on triangle meshes. Graph convolutional models were originally designed for citation graphs and other graph-structured high-level data [14], but more complex models have since developed for object meshes [28], [16]. These models are typically developed for and tested on CAD models, which allows certain design choices such as using vertex coordinates. Unfortunately coordinates change significantly with rotation or rescaling and therefore these methods are not suitable for the unpredictability of real-world data.

High performance is achieved by taking advantage of the progress made in 2D classification by using a collection of 2D views of the object. They either pool over views [23], [24], apply more advanced schemes such as intelligently clustering before pooling [29] or jointly learning the corresponding object pose [13]. Beyond the power of such representations, they also take advantage of networks pretrained on large scale 2D datasets. Once again, however, they have been designed and tested only on CAD models. Experiments suggest they depend on the outline of the object in the 2D view, which is significantly affected by occlusions [24].

Another option for 3D data is to apply 3D convolution on voxel grids [33], [17]. 3D fixed grids are, by design, very sensitive to differences in object orientation and occlusion. It is possible to train with objects in a variety of poses [22], [34], but this amounts to learning as many representations as orientations. Resultingly, the methods require a larger number of parameters to attain a given accuracy. Working with an additional dimension compared to images also leads to the parameter count increasing much faster. The trade-off between the coarseness of the grid and model complexity inherent to this representation led to the exploration of more powerful fixed 3D grid representations such as the signed distance function [11]. The limitation can also be counterbalanced by using multiple resolutions [20]. KD-tree-based models [15] push the multi-resolution idea further. Learning from this data structure achieves high accuracy but the KD-tree itself is sensitive to sensor noise and slight rotations, which makes it unsuitable for real sensor data.

Depth sensors sample perceived surfaces and provide point cloud data that is used for direct learning. PointNet [18] exploits this data type by learning from one point at a time before using a global max pooling layer to optimize globally over all the positions in the unit sphere, independently of the order of the points in the set. Research is still very active in this area with methods creating local kernels [26], [25] or exploring novel classifiers [5].

B. Robust 3D Classification

The methods mentioned until now are evaluated using the coordinates of the models in the unit sphere. Objects extracted from a reconstructed scene, however, come in any orientation, which is often detrimental to the performance of methods that ignore the difference. Common approaches to this challenge are to train over various orientations as in [17] or to use a spatial transformer layer [10] to learn an alignment of the objects as in [18].

Another direction is to learn from rotation-invariant features as is explored in [4], [30], [3]. While a new representation is introduced in [4], most work take inspiration from classical feature descriptors and explore the potential combination with deep networks. For example, [3] takes inspiration from the Point Pair Features (PPF) descriptor [7] and use convolutional neural networks to learn a multi-dimensional histogram without losing the correlation between the features. The ESF descriptor [32] is another handcrafted descriptor and is designed specifically for classification. It concatenates histograms over those sampled features, which is given to an SVM to classify. [30] combines some of the features from ESF with PPF to learn local structures that are later combined with a graph convolutional network.

III. Learning from Object Parts for Robust 3D Object Classification

Our method is developed for over- or under-segmented objects represented by manifold triangle meshes (a mesh is a manifold if each edge is connecting at most two triangles). The reason for using a mesh is that surfaces are preserved. Reconstruction methods generate that representation either directly [21] or by applying a post-processing step such as the marching cube algorithm [12]. However, the method presented here could easily be adapted for dense point clouds by using nearest neighbor approaches to retrieve the neighborhood of each point.

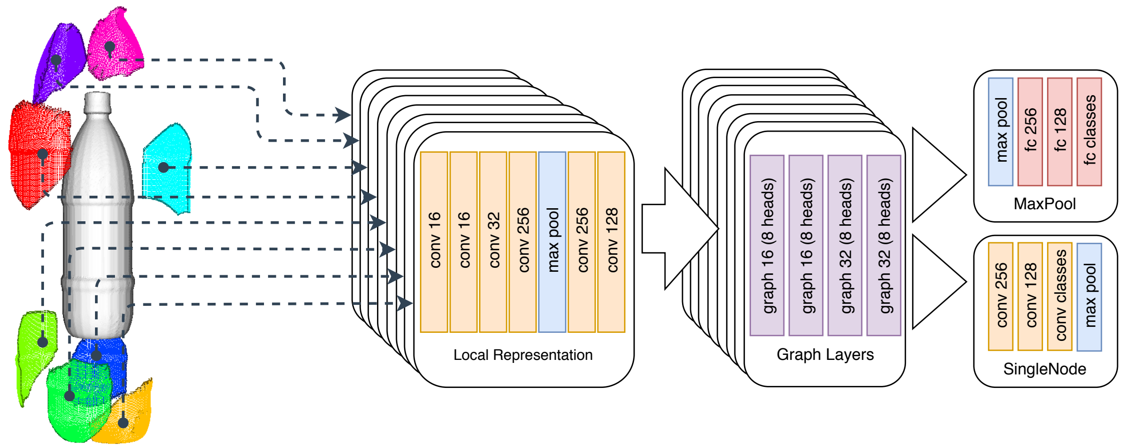

Fig. 2. Architecture overview of our proposed method. The number of parts is reduced to eight for readability purposes and the connectivity is not displayed. Convolution layers all consider only a single element (kernel size one) and the number of filters is indicated.

This section describes the proposed method. We first explain the object part sampling process and the part representation. We then outline the learning approach for the graph of object parts and the design of our graph convolution architecture.

A. *Creating object parts*

1) Object parts sampling: To transfer from artificial object models to real reconstructed data, object parts should be repeatable under varying orientation, occlusions, scale and point density. Scale-invariance forbids the use of Euclidean distance for sampling parts. To avoid sampling a part that would span through an object, and thus being significantly more sensitive to occlusions, parts are grown by following the surface of the object. The average angle between a triangle and its neighbors on a surface is used when deciding whether a neighboring triangle should be added to the part. Since reconstruction algorithms account for sensor noise and artificial data does not suffer from any random noise, high quality normals can be computed for both type of data. The angle between two neighboring normals is independent of scale and orientation of the object, and in a perfectly noiseless case, even independent of the surface sampling density. Object parts are then extracted by performing a Breadth-First Search (BFS) on the graph defined by the object mesh, or in other words, incrementally adding a one-ring neighborhood around the sampled part center. Due to the strong unpredictability of occlusions, part centers are randomly sampled. Centers are sampled so long as they do not belong to a previously sampled part. The search is stopped when the accumulated angle over the object part reaches a set threshold. The accumulated angle is computed from the average angle of each triangle, which is simply the average of the angle with each triangle neighbor.

Reconstructed scenes do not provide perfectly smooth surfaces, therefore, we perform low-pass filtering on the normals defined by the triangles. Normals for all points are first computed by averaging the normals of each triangle they belong to. Then triangle normals are computed by averaging the normals of the three points. The resulting normals are smoother than the original mesh.

2) Object part features: The object part representation should maintain the properties of the sampling. In this work, we sample a fixed number of points from the object part to generate a fixed size representation from parts of varying sizes. Orientation-invariance is then achieved by defining a local reference frame (LRF). The center of the LRF is defined by the mean of a set of points and we propose two different orientations.

The first design option is to perform Principal Component Analysis (PCA) with the set of points and use the eigenvectors as the LRF. The first and last eigenvectors (when ordered by decreasing eigenvalues) are kept and the direction of the last eigenvector is flipped in order to follow the average direction of the surface normals of the set of points. This guarantees a different LRF for concave and convex sets. The last vector is the cross-product of the first two vectors. This LRF provides a total orientation invariance and is referred to as PCA-LRF.

The second option is to define the LRF based on the global vertical axis (Z-axis) and the component of the mean surface normal of the part that is orthogonal. This is no longer independent of the orientation of the object but only independent of the orientation of the object around the Z-axis. This local frame of reference is referred to as Z-LRF. Although it is only partially orientation-invariant, it offers a more informative representation. Since many objects have a small number of canonical poses (e.g. most bottles stand upright), it remains beneficial when tested on realistic data.

All point sets are rescaled to the unit sphere to make the representation independent of the scale. In most experiments in Section IV, we also add the average angle value as a feature to the point coordinates. When sampling the point, we use the average angle value of the triangle it belongs to. It slightly improves the accuracy without any extra computation overhead because it is computed during sampling.

Finally, the graph is constructed by connecting parts that overlap. In other words, parts are connected if at least one triangle in each of the parts was sampled in the original object.

B. Model Architecture

1) General architecture: The architecture, as shown in Figure 2, follows the PointNet model where each point extracted from a part is independently fed to the same convolutional neural network. In difference to the architecture of PointNet, the spatial transformer network is unnecessary as the points' coordinates are already defined in a LRF. Each of the layers includes a batch normalization step [9]. A weighted version of the maximum value (over the whole set) of a given filter is subtracted from each output as described in [19].

The proposed model includes graph convolution layers inspired by the GCN model introduced in [14]. This is a simplification of larger graph convolutional models because only first-degree neighbors are considered. As a result, the model cannot differentiate neighbors from each other. To address this, we introduce an attention model in our graph convolutions.

2) Attention model: The GCN model is made more powerful by introducing an attention mechanism. Instead of considering all neighbors as equal, the neighbors are weighted according to a criterion that is specific to the attention model chosen. Examples of attention models have been developed in [28] and [27]. To further improve the representational power of a network, multiple attention heads are used for the same layer, and the attention heads output are concatenated at each layer.

Introducing attention to the GCN model amounts to learning a valid coefficient to replace the normalization factor. We follow the model defined in [27] where the graph convolution layer becomes

$$h_{v_i}^{l+1} = \sigma\left(\sum_{j\in\mathcal{N}_i} \gamma_{ij} h_{v_j}^{l} W^{l}\right), \tag{1}$$

where h_{v_i} is the feature vector of the i-th vertex, σ is the activation function, and W is the parameter vector of the layer l. The coefficient γ_{ij} is defined as

$$\begin{aligned}\gamma_{ij} &= \text{softmax}(e_{ij}),\\ &= \frac{\exp(e_{ij})}{\sum_{k\in\mathcal{N}_i}\exp(e_{ik})},\end{aligned} \tag{2}$$

where $e_{ij} = \text{LeakyReLU}(a^T.[h_{v_i}W || h_{v_j}W]))$, $(\cdot)^T$ denotes the transpose operation, $\cdot||\cdot$ denotes the concatenation operation and a is the vector of learned parameters for the attention. In order to respect the part connectivity, we add a bias matrix to the e_{ij} term before applying the softmax in which disconnected nodes have a value of -10^9 and connected nodes have a value of 0. This model is further discussed in [27] and [31].

3) Summarizing over object parts: In this work, we are interested in predicting the object-level class. The object part representation described so far affords a number of different options for this task. The most straightforward option is to simply perform a max-pooling operation on the feature vectors of each part and then classifying the object (referred to as MaxPool). However, the classifier will be trained expecting all nodes and is therefore less likely to transfer well to real reconstructed data that have missing nodes. A second option is to predict one class per node and average all predictions into an object-level prediction (referred to as SingleNode). Both options are evaluated in Section IV. The single node prediction trained on artificial data still provides a representation that assumes perfect connectivity. We therefore propose one last option in which a proportion of nodes are randomly disconnected (except for self-connections).

IV. Experiments

This section presents the experimental results. The first set of experiments compares our proposed approach to state-of-the-art methods for object classification on artificial data using the ModelNet dataset [33]. The second set of experiments evaluates the transfer abilities from artificial data (ModelNet) to real-world data (objects extracted from the ScanNet dataset [6]). We also evaluate in more depth the impact of the object part size and the connectivity of the object parts graph. Lastly, our method is evaluated against the PointNet architecture when training and testing on real-world data with objects extracted from the ScanNet dataset.

A. Experimental setup

1) Implementation: Our final model is described in Figure 2. We sample up to 32 parts per object and 250 points per part. The representation of each object part is fed through four 1D convolutional layers and max-pooling is applied over the whole set of points from the object part. The local representation is then passed to four graph convolutional layers. The output of each attention head is concatenated at each layer before being fed to the next. When predicting over single nodes or over a set of nodes, pooling is applied either before or after the classification layers.

2) Datasets: Evaluation is performed on two datasets: The ModelNet dataset [33] and the ScanNet dataset [6] (1513 reconstructed rooms). We use both ModelNet40 (12311 CAD models split between 40 classes) and the ModelNet10 subset (4899 models in 10 of the original 40 classes: `bathtub`, `bed`, `chair`, `desk`, `dresser`, `monitor`, `night_stand`, `sofa`, `table`, `toilet`). The second version of the annotation for ScanNet is used with the train/test/val split defined in the first version. Objects are extracted according to the annotation in the dataset. Afterwards the object classes are mapped to the ModelNet classes. The ModelNet10 mapping is the subset of ScanNet objects that belong to the ModelNet10 classes. The ModelNet40 mapping is the same but uses the ModelNet40 classes.

TABLE I

CLASSIFICATION ACCURACY ON THE MODELNET40 DATASET [33] (**MN40REC** INDICATES THAT THE METHOD WAS EVALUATED ON THE RECONSTRUCTED MODELS)

Method	MN40	MN40Rec	Input
VoxNet [17]	83.0	-	Voxel Grid
KD-Networks [15]	91.8	-	KD-Tree
MVCNN [23]	90.1	-	Views
MVCNN-New [24]	**95.0**	91.7	Views
3DmFV-Net [2]	91.6	91.1	Point Cloud
3DCapsules [5]	92.7	-	Point Cloud
PointNet [18]	89.2	88.1	Point Cloud
*Ours (PCA-LRF)	-	86.9	Mesh
*Ours (Z-LRF)	-	89.4	Mesh

B. Evaluation on artificial data

Table I compares the performance of our method to state-of-the-art methods on the ModelNet [33] dataset. It should be noted that the models of the dataset are non-manifold meshes. To apply our method, we project views of the objects and reconstruct them using a TSDF-based method. As a result, the object models differ slightly. For reference, we provide the accuracy of available methods on both versions and observe a drop in accuracy between one and three percent for the object models created for our approach.

This experiment shows that despite being specifically designed with real-world constraints in mind, our method still shows competitive results on artificial data. The results presented in Table I correspond to our method with a max pooling and trained with the average angle values as an included feature. Table II shows the results of the addition of the average angle value as an extra feature. We see that performance slightly improves in all conditions without adding any computation as it is already calculated during the sampling process. Furthermore, the SingleNode type of pooling (i.e. predicting a class for each object part and averaging the prediction over the object) gives similar results to MaxPool on artificial data. An experiment using the GCN model [14] without the attention model is also performed. This result shows that the attention model improves the results, which is consistent with [27]. All future results include the average angle as a feature, use the Z-LRF and include the attention model.

Figure 3 illustrates the benefits of our design when simulating levels of occlusion on artificial object. The occlusion is simulated by removing a region that is grown on the mesh surface until a set percentage of the object is reached. We observe that the performance degrades significantly better than for the PointNet architecture.

C. Evaluation of the gap between real and artificial data

This section evaluates the gap when training on artificial data (objects from ModelNet) and testing on real-world data (segmented objects from ScanNet). This is a difficult task because of the domain shift as well as some important characteristics of the ScanNet dataset (see Figure 4). ScanNet was first and foremost designed for semantic segmentation, which is more concerned with large-scale structures. Additionally, since it represents natural environments, the dataset is strongly imbalanced. Chair is a highly dominating class, which is seen by the performance of the "chair predictor" baseline (i.e. always predicting the chair class) in Table III. As such, class accuracy is a more relevant metric than overall accuracy. Moreover, most structures tend to be oversmoothed by the reconstruction algorithm. This is a side-effect of the reconstruction algorithm that tries to reconcile various noisy measurements. Subtle differences also exist between ScanNet classes and ModelNet classes (e.g. a pack of bottles in ScanNet is mapped to the bottle class, whereas that class only contains single standing bottle in ModelNet). Finally, the segmentation is often inaccurate for smaller objects. This is due to the fact that scenes are oversegmented and then clusters are annotated. Therefore, many extracted objects include points from the surrounding elements. All these factors make this a very challenging dataset for the task.

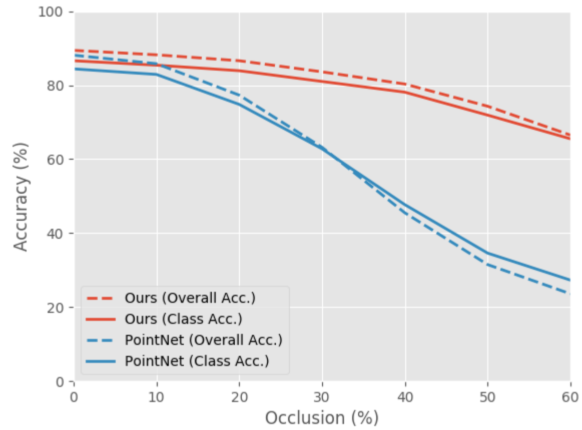

Fig. 3. Accuracy in percent for various simulated occlusion (ModelNet40 is used, SingleNode model trained with 75% Disconnect).

TABLE II

EVALUATION OF DESIGN CHOICES ON MODELNET10

Pooling	LRF	Avg ang.	Attention	Acc.
MaxPool	PCA-LRF	✗	✓	85.6
MaxPool	PCA-LRF	✓	✓	86.3
MaxPool	Z-LRF	✗	✓	87.2
MaxPool	Z-LRF	✓	✗	86.9
MaxPool	Z-LRF	✓	✓	89.2
SingleNode	Z-LRF	✓	✓	89.6

As shown in Table III, only MVCNN-New and our model manage to perform some transfer. The result when always predicting the chair class not only demonstrates the imbalance of the dataset but also provides a general reference for evaluating performance. Considering that our object part representation could easily be swapped out in our pipeline, a comparison to PointNet is fairest to our contribution. This local representation was selected for its versatility and proven

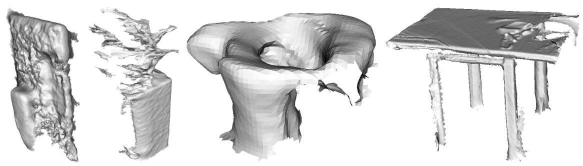

Fig. 4. Illustration of the noisy objects extracted from the ScanNet dataset. Left to right: monitor (very noisy surface), potted plant (strong occlusions and many disconnected parts), cup (small object) and table (object on top merged with it).

TABLE III

EVALUATION WHEN TRANSFERRING FROM MODELNET40 TO SCANNET

	Acc.	Cls Acc.
Chair Predictor	36.2	2.5
PointNet	2.2	3.3
3DmFV-Net	0.9	4.2
MVCNN-New	**36.6**	**20.0**
Ours	34.6	19.1

reliability in various contexts [1], [8]. The good results of the MVCNN-New architecture supports the idea that part-based (in this case parts are defined by projective geometry), and invariance to rotation around the gravity axis is essential to transfer to real reconstruction. MVCNN-New seems to work best if one of the sampled view preserves the outer contour of the reconstructed object. In particular, MVCNN-New is better than our method on the classes `plant`, `sofa` and `toilet`, but worse for `bathtub monitor` and `sink`. It is interesting to note that both sink and bathtub are both box-like objects with concave parts that do not affect the outer contour of the object in a projected view. Our model is adversely affected by meshes having many disconnected components such as `plant` and `bookshelf` meshes in the ScanNet dataset.

For our method, the best results are achieved by increasing the sampling threshold by a factor of two compared to training. Also, SingleNode pooling is used with a disconnection rate of 75% during training. In the next section, we evaluate those design choices in more detail.

1) Influence of the part size in the transfer: One significant parameter in the transfer performance is the threshold set for the part sampling algorithm. ScanNet scenes tend to be oversmoothed, which affects the angle-based sampling procedure for objects that have large flat surfaces in artificial models. Also, scenes are reconstructed at a constant density, which means that smaller objects have a smaller density than bigger objects. Combined with the low level of noise, the right trade-off needs to be found for increasing the threshold used in training and testing because classes react differently. Figure 5 shows the impact of the sampling threshold on the accuracy when using the mapping from ModelNet40 classes to the ScanNet objects. The best threshold typically increases with the increase of the average size of the class. For small objects, such as `bowl`, the best threshold is close to the training value. For medium-sized objects, such as `toilet`, the best threshold is between three and fours times

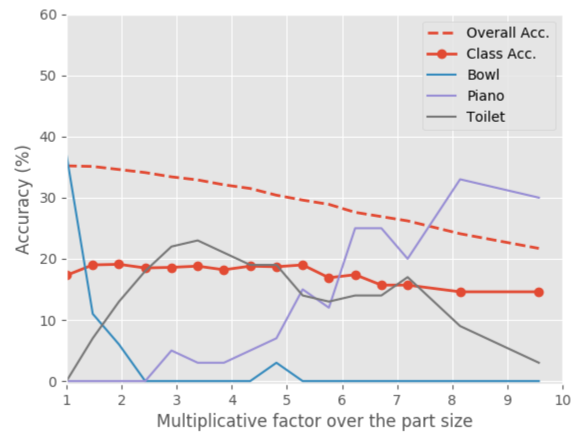

Fig. 5. Accuracy in percent for various multiplicative factors applied to the sampling threshold (ModelNet40 mapping is used, SingleNode model trained with 75% Disconnect).

TABLE IV

EVALUATION OF DESIGN CHOICES FOR TRANSFERRING TO SCANNET (USING THE MODELNET10 MAPPING)

Pooling	Disconnect (%)	Acc.	Cls Acc.
MaxPool	-	44.1	42.35
SingleNode	0	50.1	38.3
SingleNode	50	62.2	39.0
SingleNode	75	**62.9**	**43.2**
SingleNode	100	54.5	33.2
Chair predictor	-	56.0	10.0

the original value. For very large objects, such as `piano`, the best best threshold is up to eight times. Larger objects simply have more triangles in the mesh. The threshold is reached much faster due to the noise, therefore, they require a larger threshold in order to reproduce parts similar to those observed during training.

2) Influence of the connectivity and pooling: Table IV shows the impact of different pooling strategies on the transfer performance. It also shows the impact of randomly disconnecting some object parts in the object part graph during training in order to better approximate occluded objects. The experiments are performed using the mapping from ModelNet10 to ScanNet. The significant increase in accuracy is due to the fact that the ModelNet10 mapping contains most of the well-reconstructed objects because classes of ModelNet10 corresponds to objects of larger scale (excluding classes such as `cup`, `keyboard`, `laptop`, `vase` and `xbox`). Clearly, performance improves with disconnection considered. Applying 0% has low accuracy because it does not model the occlusion. At 100% the performance is also low because this over-estimates the level of occlusion. The best compromise is achieved with 75% for the ScanNet dataset. The disconnect experiments are not applied to the MaxPool setup, as it would still take into account each part when max-pooling, thus failing to simulate occlusions.

TABLE V

EVALUATION WHEN TRAINING AND TESTING ON SCANNET (USING THE MODELNET40 MAPPING)

	Acc.	Cls Acc.
Chair Predictor	36.2	2.5
PointNet	73.6	52.3
PointNet (rescaled obj.)	53.4	17.2
Ours (Z-LRF)	**85.2**	**62.8**

D. Evaluation on real data

The large size of ScanNet makes it feasible to train methods on real-world data instead of artificial data. This is helpful because it can establish an upper bound for the transfer to this dataset. The results in Table V are significantly higher than when transferring, which implies the existence of occlusion consistency for a given class. We, however, conjecture that this only holds for larger structures but not for smaller objects, such as household items. The results additionally show that an advantageous side-effect of our design is the ability to better learn from noisy data. Our method achieves an accuracy of 85.2% and class accuracy of 62.8%. The best results are achieved using the Z-LRF frame of reference. Prediction is performed for each node and averaged, and training is performed with a 75% random chance of disconnecting two neighbors. In comparison, the PointNet architecture achieves much lower scores of 73.6% accuracy and 52.3% class accuracy, which is approximately 10% less than our method. Experiments are also performed with PointNet trained on objects rescaled to the unit sphere to prevent the method from taking advantage of scale information. The significant decrease in this case suggests that PointNet finds it more difficult to establish consistent shapes. This supports our argument that even though objects have inconsistent shapes, they always have consistent parts.

V. CONCLUSION

This paper addressed the important robotics task of object classification from 3D data. We present an approach that transfers to real reconstructed objects when trained on clean CAD models only. Results show that our approach significantly outperforms state-of-the-art methods while maintaining competitive performance when training and testing on CAD models. The performance increase in the transfer is achieved through sampling object parts that are reproducible under rotation, occlusion and scale in combination with a graph-based deep learning architecture. Learning with a rotation-invariant and scale-invariant representation of parts enables objects to be recognized with significant portions of missing data.

The next step for this work is to create a large-scale dataset focused on objects rather than scenes to support further research in the task of transferring from widely available CAD models to noisy real-world data. This will be particularly important for the task of retrieving CAD models from candidate segments in noisy data. This will be highly relevant for service robotics systems that have to prepare for a variety of contexts by building a general world knowledge when deployed in user homes.

REFERENCES

[1] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "Pointnetlk: Robust & efficient point cloud registration using pointnet," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[2] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3145–3152, 2018.

[3] D. Bobkov, S. Chen, R. Jian, M. Z. Iqbal, and E. Steinbach, "Noise-resistant deep learning for object classification in three-dimensional point clouds using a point pair descriptor," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 865–872, 2018.

[4] C. Chen, G. Li, R. Xu, T. Chen, M. Wang, and L. Lin, "Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[5] A. Cheraghian and L. Petersson, "3dcapsule: Extending the capsule architecture to classify 3d point clouds," *CoRR*, vol. abs/1811.02191, 2018. [Online]. Available: http://arxiv.org/abs/1811.02191

[6] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5828–5839.

[7] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 998–1005.

[8] L. Ge, Y. Cai, J. Weng, and J. Yuan, "Hand pointnet: 3d hand pose estimation using point sets," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of International Conference on Machine Learning*, 2015, pp. 448–456.

[10] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proceedings of Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[11] J. Joon Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[12] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-time large-scale dense 3d reconstruction with loop closure," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII*, 2016, pp. 500–516.

[13] A. Kanezaki, Y. Matsushita, and Y. Nishida, "Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of International Conference on Learning Representations*, 2017.

[15] R. Klokov and V. Lempitsky, "Escape from cells: Deep KD-Networks for the recognition of 3D point cloud models," in *Proceedings of IEEE International Conference on Computer Vision*, 2017, pp. 863–872.

[16] I. Kostrikov, Z. Jiang, D. Panozzo, D. Zorin, and J. Bruna, "Surface networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Oral Presentation)*, June 2018.

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

[18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[19] S. Ravanbakhsh, H. Su, J. Schneider, and B. Poczos, "Deep learning with sets and point clouds," in *Proceedings of International Conference on Learning Representations*, 2017.

[20] G. Riegler, A. Osman Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3577–3586.

[21] S. Schreiberhuber, J. Prankl, T. Patten, and M. Vincze, "Scalable-fusion: High-resolution mesh-based real-time 3D reconstruction," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2019, pp. 140–146.

[22] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox, "Orientation-boosted voxel nets for 3D object recognition," in *Proceedings of British Machine Vision Conference*, 2017.

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

[24] J.-C. Su, M. Gadelha, R. Wang, and S. Maji, "A deeper look at 3d shape classifiers," in *Computer Vision – ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds. Cham: Springer International Publishing, 2019, pp. 645–661.

[25] M. Tatarchenko*, J. Park*, V. Koltun, and Q.-Y. Zhou., "Tangent convolutions for dense prediction in 3D," *CVPR*, 2018.

[26] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," *arXiv preprint arXiv:1904.08889*, 2019.

[27] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018, (accepted as poster).

[28] N. Verma, E. Boyer, and J. Verbeek, "Feastnet: Feature-steered graph convolutions for 3d shape analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2598–2606.

[29] C. Wang, M. Pelillo, and K. Siddiqi, "Dominant set clustering and pooling for multi-view 3d object recognition," in *Proceedings of British Machine Vision Conference (BMVC)*, vol. 12, 2017.

[30] J. Weibel, T. Patten, and M. Vincze, "Robust 3d object classification by combining point pair features and graph convolution," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 7262–7268.

[31] J.-B. Weibel, T. Patten, and M. Vincze, "Geometric priors from robot vision in deep networks for 3D object classification," in *Proceedings of International Conference on Robotics and Automation (Workshop on Multimodal Robot Perception: Perception, Inference and Learning for Joint Semantic, Geometric, and Physical Understanding)*, 2018.

[32] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification," in *Proceedings of IEEE International Conference on Robotics and Biomimetics*, 2011, pp. 2987–2992.

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

[34] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1802–1811.