Sim2real Transfer Learning for Point Cloud Segmentation: An Industrial Application Case on Autonomous Disassembly

Chengzhi Wu[1], Xuelei Bi[1], Julius Pfrommer[2,3],
Alexander Cebulla[1], Simon Mangold[4], and Jürgen Beyerer[2]

[1]Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Germany
[2]Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB, Germany
[3]Fraunhofer Center for Machine Learning, Germany
[4]wbk Institute of Production Science, Karlsruhe Institute of Technology, Germany

chengzhi.wu@kit.edu xuelei.bi@student.kit.edu julius.pfrommer@iosb.fraunhofer.de
alexander.cebulla@kit.edu simon.mangold@kit.edu juergen.beyerer@iosb.fraunhofer.de

Abstract

On robotics computer vision tasks, generating and annotating large amounts of data from real-world for the use of deep learning-based approaches is often difficult or even impossible. A common strategy for solving this problem is to apply simulation-to-reality (sim2real) approaches with the help of simulated scenes. While the majority of current robotics vision sim2real work focuses on image data, we present an industrial application case that uses sim2real transfer learning for point cloud data. We provide insights on how to generate and process synthetic point cloud data in order to achieve better performance when the learned model is transferred to real-world data. The issue of imbalanced learning is investigated using multiple strategies. A novel patch-based attention network is proposed additionally to tackle this problem.

1. Introduction

Due to the rapid development of neural network algorithms, an increasing number of industrial companies and factories have started using deep learning (DL) methods for a variety of manufacturing and remanufacturing tasks in the past decade. In general, neural networks require a substantial amount of data in order to be trained, whereas for practical industrial applications, allocating and annotating a large amount of data is difficult or even impossible, especially when robots are involved. In the field of robotics, when the robot or manipulator directly interacts and samples with the real-world environment, there will be problems of low sampling efficiency and safety problems.

One possible solution to this problem is to apply simulation-to-reality (sim2real) method, which learns with simulated data and transfers the learned knowledge to real-world application. This is a common strategy used in robotics for learning robot movement controls [2, 33] and robotic-related vision tasks [42, 24, 29]. In those computer vision tasks, simulated scenes are usually rendered into RGB images with possible auxiliary depth, thermal, or even flow images. Then, DL-based neural networks are pre-trained with the synthetic data and subsequently transferred to real-world use cases via domain adaption. However, most current sim2real work focuses on image data. Few researches apply sim2real methods on point cloud data. In this paper, we show a full pipeline of how to perform sim2real transfer learning on point clouds for a robotics use case as a part of a practical remanufactoring application.

We consider the automated disassembly of different variants of actuators which are commonly used in vehicle manufacturing, *e.g.*, as seat adjuster motors, window lift motors or rear door motors. Several example motors are shown in Figure 1(a). The ultimate goal of this project is to use robots to perform automatic disassembly of motors, not only for the known motor types, but also for future variants with unseen specifications. In this case, generating a synthetic dataset with motor variants in simulated scenes for sim2real transfer learning [55] is a good solution. By learning the internal structure on part level, (*e.g.* gear container, pole pot, electrical connection), processes on unseen variants which have similarities to the known population of actuators become feasible. This paper focuses on the first step of getting precise screw positions and orientations on motor covers for robots as one of the most important tasks for disassembly.

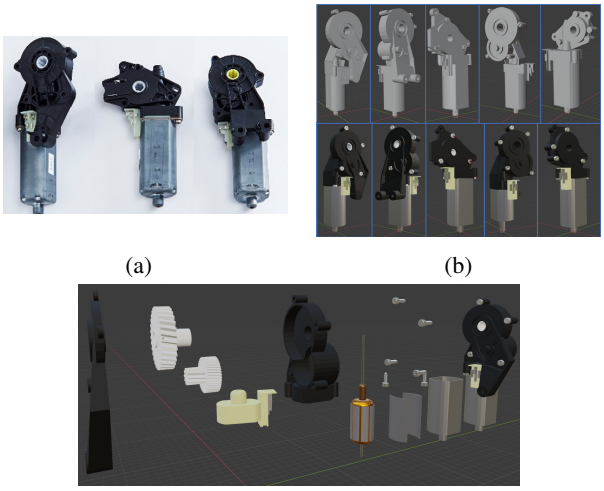


Figure 1: Real-world motors and generated demo motors. (b) Upper row: no textures added; bottom row: textures added and rendered. (c) An explosion figure of a generated motor. The original assembled motor model is also shown at the right most.

Generating a synthetic point cloud dataset for sim2real transfer learning has following advantages in our project: (i) a large synthetic dataset can be easily created, segmentation ground truth labels are given in the simulation, no manual annotation needed; (ii) motor variants with unseen specifications may be generated, which will strengthen the generalization ability of the trained network model; (iii) point cloud data contain richer 3D information for the learning. Using point cloud data avoids some problems that may occur when using image data, *e.g*., colors of the simulated images are far from realistic since it is hard to get the perfect textures for scene objects or to render the scene with perfect lighting conditions. When using the point cloud dataset, we use point coordinates information other than colors.

The remainder of this paper is structured as follows: Section 2 summarizes the state-of-the-art of 3D synthetic dataset creation, sim2real transfer learning, and point cloud segmentation. Section 3 shows a general pipeline of creating a synthetic dataset with simulated scenes. Section 4 describes the whole sim2real learning framework and gives experimental results. Section 5 additionally explores several strategies for imbalanced learning, including a novel patch-based attention network module. Finally, Section 6 summarizes presented results and discusses future work.

2. Related Work

3D synthetic dataset. Generating synthetic datasets as training data for machine learning purposes has already been widely discussed and used as a learning approach for various computer vision applications. In the past decade, many synthetic datasets of 3D models have been created, including the Princeton Shape Benchmark [38], ModelNet [50], ShapeNet [6], PartNet [28], etc. They collect large amounts of 3D models of different categories. A large dataset of 3D-printing models is provided in Thingi10K [54], while a more recent ABC dataset [19] collects over 1 million CAD models including many mechanical components. Regarding 3D scenes, [45], [36] and [18] generate synthetic datasets for the segmentation and detection of objects in virtual urban scenes. [22] generates images from virtual garden scenes, while [43] creates a dataset for pose estimation. There are also works that generate synthetic point clouds. SynthCity [12] generates point clouds of urban scenes using Blender, while [30] also uses Blender but for the generation of point clouds of historical objects.

Sim2real transfer learning. By allowing faster, more scalable, and lower-cost data collection than is possible in real-world, sim2real approaches show great impact on machine learning and have been applied in many fields including robotics and classic machine vision tasks. [42], [24] and [44] train neural network models on synthetic RGB images with domain randomization or domain adaption then transfer it to real-world, while Pachevish *et al*. [29] work with synthetic depth images. Also working with synthetic image data, Du *et al*. [9] propose a method for automatically tuning simulator system parameters to match the real world. With the help of deep reinforcement learning [27], robotics policies are directly used as training data for sim2real learning in some works [25, 2]. A more detailed survey is given in [53]. Apart from robotics tasks, sim2real methods have also been widely used in other fields including autonomous driving [51, 34], medical diagnosis [1], or even the control of atmospheric pressure plasma jets [48].

Point cloud segmentation. Before the appearance of PointNet [31], deep learning-based methods for point cloud segmentation are usually multi-view based [20, 5, 3, 40] or volumetric-based [26, 17, 21]. PointNet [31] is the first DL-based method that learns directly on points. It uses point-wise multi-layer perceptrons to extract global features. Its subsequent work of PointNet++ [32] further considers local information. PointConv[49] and KPConv [41] propose point-wise convolution operators with which points are convoluted with neighbor points. Similar ideas are proposed in [46, 16]. Simonovsky *et al*. [39] takes each point as a graph vertex and applies graph convolution. In DGCNN [47], EdgeConv blocks update the neighbor information dynamically. RandLA-Net [15] learns attention scores for points as a soft mask to replace the original pooling layer. GAPNet [7] and Liang *et al*. [23] propose graph-attention operations with neighbor points to learn coefficients. More recently, transformer-based methods are starting to trend. PCT [14]

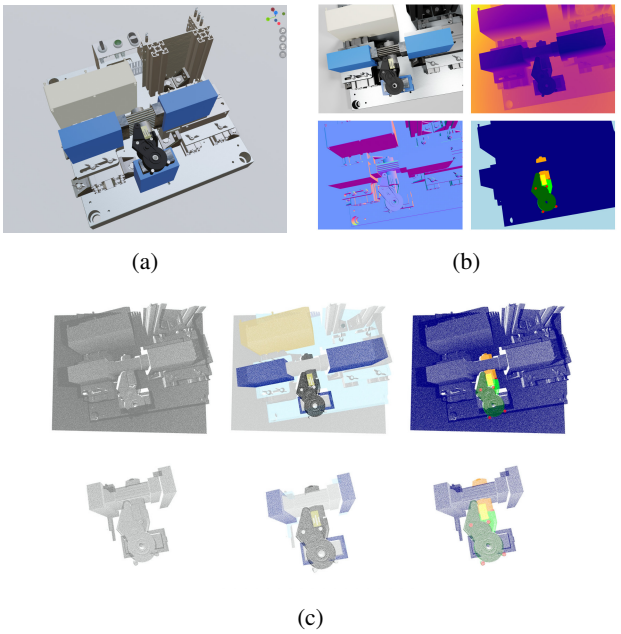


Figure 2: Synthetic dataset generation: (a) simulated scene built in Blender; (b) synthetic image data generated with BlenderProc; (c) synthetic point cloud data generated with BlenSor.

pioneers on this direction by replacing the encoder layers in the original PointNet [31] framework with self-attention layers, while PT [52] is based on U-Net [35]. SortNet is proposed in [10] to learn sub-point clouds, with which attention operations are applied on their latent features and the global feature to perform local-global attention.

3. Synthetic Dataset Generation

3.1. Synthetic Mesh Model Generation

To easily generate motor mesh models of a variety of specifications, we create a Blender addon based on the motor types we have. As an open source software, Blender [4] is a proven tool that performs well in modeling shapes and creating highly customizable addons. Our addon is able to generate motor mesh models with various specifications and save them in desired file formats. Each component of a generated motor can also be saved separately.

The generated models contain the following components: (i) Pole Pot; (ii) Electric Connection; (iii) Gear Container; (iv) Cover and (v) Screws. Those are the five main categories we need perform segmentation on. Additionally, following inner components have also been generated : (vi) Magnets; (vii) Armature; (viii) Lower Gear and (ix) Upper Gear as presented in 1(c). However, since we only focus on the first step of unscrewing process in this paper, inner parts will not be investigated. To generate motors with various specifications, we provide lots of parameter options that control the type, size, position and rotation of different parts of motor, *e.g.* screw position, gear size, or pole pot length. Figure 1(b) shows ten generated demo motors with different parameters and an exploded view of a demo motor. All the individual components mentioned above are modeled separately as illustrated.

3.2. Synthetic Point Cloud Generation

The generated mesh models are further used to create synthetic image and point cloud datasets. A simulated scene is built in Blender for it. Apart from the lights and cameras, the Blender scene also contains the model of the real-world clamping system and a background panel. The camera rotates randomly on top of the scene within a certain range yet always towards the motor. To create image dataset, apart from the scene images rendered by Blender directly, BlenderProc [8] can be used to generate corresponding depth images, normal images, and segmentation ground truth images as shown in Figure 2(b). Or in our case, BlenSor [13] is used to simulate the sensors to create point cloud dataset. Figure 2(c) gives a demo of generated point cloud colored in the material color or its segmentation ground truth.

3.3. Data Pre-processing and Augmentation

The generated point clouds are of relatively large size. Each point cloud contains around 1.2 million points. However, neural networks have a limitation of point number per batch, common choices are 1024/2048/4096 points. Direct downsampling makes the sub-point clouds contain too less points for tail categories (*e.g.*, screws in our case) thus does not work. To deal with large point clouds, common industrial applications use sliding voxels to voxelize the point cloud space and perform prediction voxel-wise. In our case, we want to perform the point cloud segmentation fast and precisely to prevent the robot from being idle for a too long time. Using the prior knowledge that motors are always clamped in a relatively fixed position in the clamping system and only the area around the motor is of our interest, we restrict the sampling region by cropping a cuboid area around that location and use it as input to the segmentation neural network. All other residual points are labeled as background points directly. The size of the cropped point cuboid is of only around 10% of the raw point cloud. By doing so, direct downsampling the point cuboid into sub-point clouds of 2048 points makes the tail categories still have enough points for learning. A cropped point cuboid demo is given in Figure 2(c).

Apart from pre-processing, augmentation is also an important part to improve the generalization ability of the models. Common augmentation methods include random rotation and random jittering over the whole point cloud.

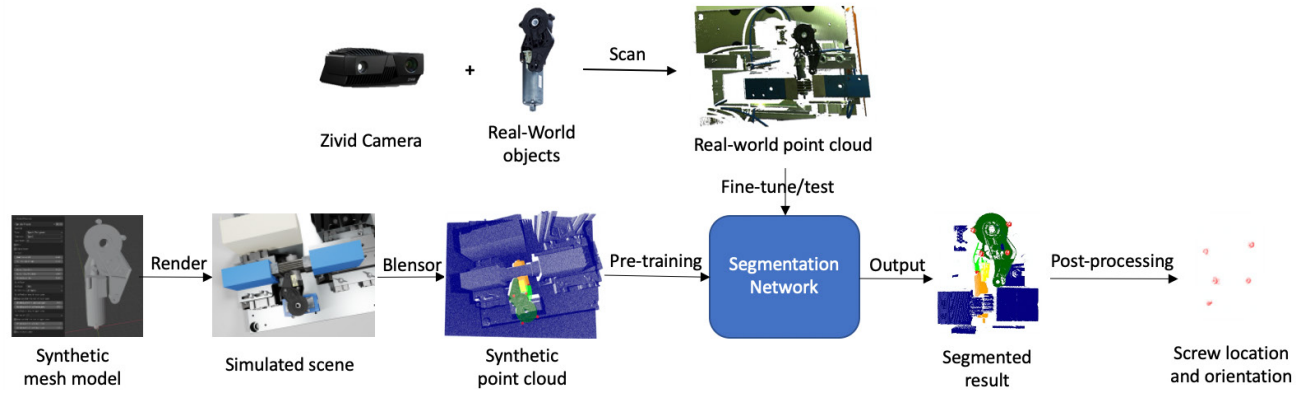


Figure 3: Sim2real transfer learning piepline for the point cloud segmentation task in our industrial application.

In our case, we additionally introduce other augmentation methods of (i) random cuboid size along all three axes; (ii) random mild translation and rotation of motors; and (iii) adding random size and random position hovering tiles over the clamping system as scene masks, which is similar to the masking augmentation on images. An detailed ablation study regarding the augmentations is given in Section 4.5.

4. Sim2real Point Cloud Learning

As illustrated in Figure 3, apart from the synthetic point cloud dataset generation, the whole pipeline consists following other steps: pre-train the network model on synthetic data, fine-tune the network model on real-world data, and post-processing for screw information.

4.1. Pre-training on Simulated Scenes

As reviewed in Section 2, there are a variety of neural network models that can be used for the point cloud segmentation task. Since the backbone itself is not of our main focus, balancing the performance and the computation time, we use DGCNN [47] as our backbone network for the segmentation task. To better deal with the camera perspective variance, a spatial transform network (STN) [31] is introduced at the beginning.

In the previous step, a dataset of 1000 random scenes with 1000 random motors is generated. We use 80% of it for training and 20% for test. In the pre-training process, the training takes 100 epochs and the batch size is 16. We use an initial learning rate of 0.01, and a *cos* decay scheduler with a final learning rate of 0.00001. The optimization method is stochastic gradient descent (SGD). We use the widely used cross entropy for the segmentation loss L_{seg}. An auxiliary STN rotation loss L_{rot} which describes the L2 difference between the learned rotation matrix and the ground truth (saved during the augmentation) is additionally applied with a small weight α. The total loss is defined as:

$$L_{\text{total}} = L_{\text{seg}} + \alpha L_{\text{rot}} \tag{1}$$

Table 1: Numerical results of models with different settings. The second column indicates whether the model is pre-trained on the synthetic dataset or not. Both results in the pre-training step (Simulation) and the fine-tuning step (Reality) are given. Note that the results in Simulation columns are the test results on synthetic test dataset, other than real-world test dataset. Same below.

STN	Pre-train	Simulation		Reality	
		mIoU	screw IoU	mIoU	screw IoU
-	-	-	-	0.8707	0.4884
✓	-	-	-	0.9030	0.6272
-	✓	0.9202	0.7120	0.9187	0.6830
✓	✓	**0.9675**	**0.8875**	**0.9375**	**0.7842**

We set $\alpha = 0.01$. During the training, the common segmentation metric, mean intersection over union (mIoU), is used to measure the model performance. The IoU of screw category is used as an additional metric since screws are of our main focus. During both pre-training and fine-tuning processes, we save the model that performs best on the screw IoU metric.

4.2. Fine-tuning on Real-world Scenes

During the fine-tuning step, we load the network parameters from the pre-training step and perform transfer learning, *i.e.*, adapting the network model from simulated scenes to real-world scenes. We took 26 real-world point clouds with Zivid camera and manually labeled them. 20 scenes are used for fine-tuning and 6 scenes are used for test. Note that this is not a small dataset since the point cloud cuboid from each scene contains around 200,000 points and can be sampled to around 100 sub-point clouds of 2048 points. In the fine-tuning process, the training takes 300 epochs and the batch size is 16. We use an initial learning rate of 0.001, and a *cos* decay scheduler with a final learning rate of 0.00001. The optimizer, loss, evaluation metrics are identical to that in the pre-training step.

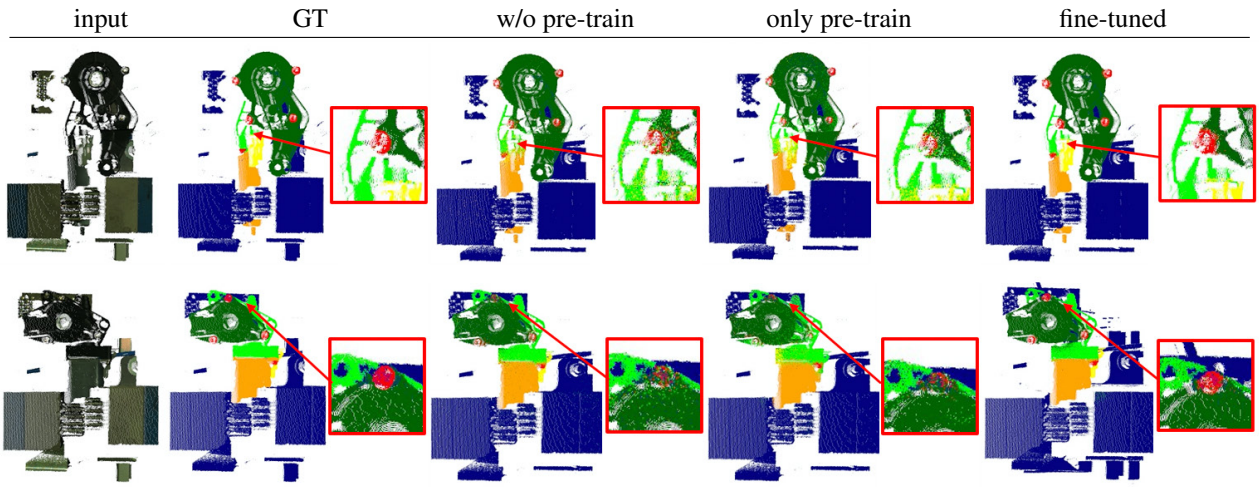


Figure 4: Segmentation result visualization on real-world test data. Without pre-train means the network is trained on real-world data directly. Only pre-train means the network is trained only on synthetic dataset and directly used for the testing on real-world data. Fine-tuned means the network is both pre-trained and fine-tuned.

Table 2: Ablation study on augmentation methods. Aug 1: random rotation and random jittering over the cuboid point cloud. Aug 2: random cuboid size along all three axes. Aug 3: random mild translation and rotation of motors. Aug 4: adding random position hovering tiles over the clamping system as scene masks.

Dataset	Augmentations				Simulation		Reality	
	aug 1	aug 2	aug 3	aug 4	mIoU	screw IoU	mIoU	screw IoU
dataset 1	✓				0.9751	0.9141	0.9280	0.7518
dataset 2	✓	✓			**0.9801**	**0.9372**	0.9368	0.7799
dataset 3	✓	✓	✓		0.9742	0.9161	0.9370	0.7815
dataset 4	✓	✓	✓	✓	0.9675	0.8875	**0.9375**	**0.7842**

Note that after the model is trained, the dataloader for the test process is different from that for the training process. When training the network, each input point cuboid splits into sub-point clouds of 2048 points as much as it can, and the residual points are discarded. When testing the trained model, the residual points are not discarded but completed into a sub-point cloud of 2048 points with other random points resampled from the original input.

4.3. Quantitative and Qualitative Results

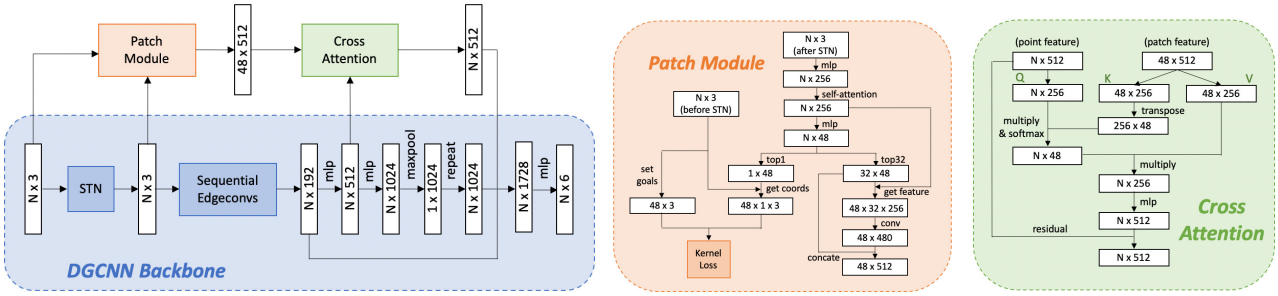
Numerical results over the metrics of mIoU and screw IoU are given in Table 1. From it, we can observe that using the STN module improves the performance drastically. On the other hand, using the sim2real transfer learning with two steps of pre-training and fine-tuning also boosts the performance. Combining both gets even better performance.

Some qualitative results are given in Figure 4. From it, we can observe that direct training on the real-world data performs decent on most points but not so good on tail categories. With pre-training on the simulated scenes and fine-tune the model on real-world scenes can achieve a much better segmentation result, especially for the tail categories.

4.4. Post-processing for Screw Information

After the network model is fine-tuned, given a real-world point cloud as input, the network outputs the segmentation result. The post-processing step aims at getting screw locations and orientations with the screw points that have been segmented out. To get screw locations, clustering algorithms are firstly used to group segmented screw points. In our case, we use the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [11, 37] algorithm for clustering. With appropriate parameter settings, each cluster is one screw. Using the prior knowledge that the bottom most screw is always the side screw, all the clusters above are cover screws that need to be unscrewed in this process. The cover screw locations are obtained by computing the center of each cluster. For the screw orientations, processing on the screw points directly is problematic since screws have uneven surfaces. Using the prior knowledge that all cover screws are having the same orientation as the flat cover, the screw orientation is actually identical to the

Figure 5: Post-processing for screw information.



(a) (b) (c)

Figure 6: Patch-based attention network. (a) Full architecture. (b) Proposed patch module, has an additional defined kernel loss, outputs patch feature. (c) Cross attention module.

normal of the cover flat part in our case. We hence apply DBSCAN on the points that are segmented as the cover category but with their estimated normals other than coordinates. With appropriate parameter settings, we can make the cluster number to be only one, which means all the points whose normal vectors are similar to the cover flat part are clustered together. Then the cover normal, or the screw orientation, is obtained by averaging all the normal vectors of those points. The process is illustrated in Figure 5.

4.5. Ablation Study on Data Augmentation

A variety of augmentation methods have been used on our synthetic dataset. To validate their effectiveness, an ablation study is performed by generating several different datasets and using a same network architecture for the pre-training and fine-tuning processes. The numerical results are given in Table 2. In the used four augmentation methods, the former two methods are augmentations performed during the data pre-processing, and the latter two methods are augmentations performed during the data generation. From Table 2, we can observe that randomly changing the cuboid size improves the segmentation performance on both pre-training and fine-tuning steps. On the other hand, while the latter two augmentation methods decreases the pre-training performance, they both improves the fine-tuning performance when the model is transferred to real-world point cloud data. In this paper, we use dataset 4 for most other experiments.

5. Imbalanced Learning

Imbalanced learning, also referred as long-tail learning in the classification tasks, is a problem where the distribution of examples across the known categories is biased or skewed. In our case, original synthetic point clouds are mostly occupied with background points (around 96%) and have extremely less screw points (around 0.1%). After applying the **sample region restriction**, *i.e.*, the cuboid crop strategy as illustrated in Figure 2(c) and described in subsection 3.3, in each point cuboid, background points take up around 64% while screw points take up around 1%. The imbalanced problem has been alleviated. Meanwhile, it is still worth investigating to further improve the performance. Several additional strategies are proposed to deal with the imbalanced learning problem in this paper. They are proposed from the perspectives of data augmentation, weighting loss, and extra network block respectively.

5.1. Focused Sampling for Tail Categories

One common strategy to deal with the long-tail problem in classification tasks is resampling like manually adding samples of the tail categories. In our case, the key category of screw only occupies a extreme small portion of the whole point cloud compared to other components, hence it is possible to increase the number of screw points by densifying them. The strategy is as follows. For each point p_1 belongs to the screw category, get its nearest same category point p_2. (a) If the nearest same category point of

Table 3: Ablation study on proposed strategies for imbalanced learning.

Sample region restriction	Focused sampling	Weighting loss	Patch-based attention	Simulation		Reality	
				mIoU	screw IoU	mIoU	screw IoU
✓	-	-	-	0.9675	0.8875	0.9375	0.7842
✓	✓	-	-	0.9627	0.8661	0.9376	0.7570
✓	-	✓	-	0.9668	0.8862	0.9409	0.7717
✓	-	-	✓	0.9693	0.9063	**0.9462**	**0.7968**
✓	✓	-	✓	0.9644	0.8850	0.9389	0.7622
✓	-	✓	✓	**0.9729**	**0.9165**	0.9412	0.7794
✓	✓	✓	✓	0.9680	0.8972	0.9401	0.7683

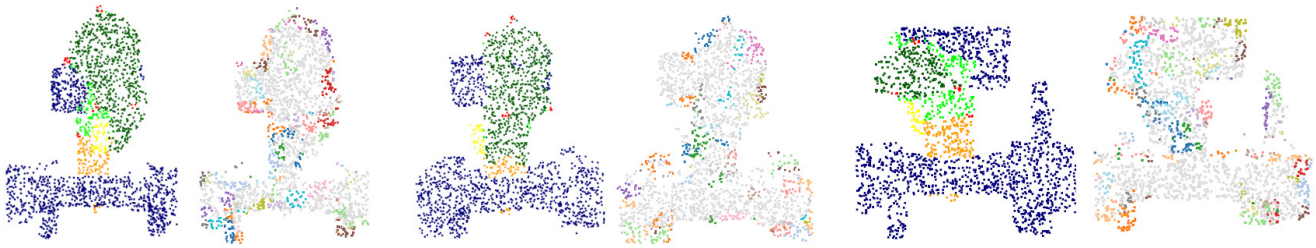


Figure 7: Visualizing learned patches on sub-point clouds of 2048 points.

p_2 is also p_1, this means both points are not at the cluster boundary, hence a new point is added with the coordinate of $p_{\text{add}} = p_1 + \frac{1}{3}(p_2 - p_1)$. (b) If the nearest same category point of p_2 is not p_1, this means p_1 is likely to be an outlier point or at the cluster boundary, hence a new point is added with the coordinate of $p_{\text{add}} = p_1 + \frac{2}{3}(p_2 - p_1)$. Above operation doubles the point number of the tail category in the training dataset.

5.2. Weighting Category Loss

Adding additional weights to each category when computing the cross entropy loss is another widely used strategy. Most current DL packages provide such an optional argument in their in-built loss functions. However, it is an unsolved question that what is the best way to compute and set the category weights. In this paper, we propose a following method. Assume the point cloud has M categories and N points in total. For each category that has $n_i (i = 1, 2, \ldots, M)$ points, its ratio is given as $r_i = n_i/N$. Then a scaled ratio sr_i is computed by decreasing the original ratio difference with a cubic root operation as $sr_i = t_i/(\sum_{i=1}^{M} t_i)$, where $t_i = (max(r_1, r_2, \ldots, r_M)/r_i)^{\frac{1}{3}}$. In this case, smaller r_i means the corresponding category gets a larger sr_i. However, using sr_i directly leads to a huge decrease on the loss magnitude. To eliminate the possible problem caused by it, an additional factor is computed as $f = \sum_{i=1}^{M} r_i \times sr_i$ and multiplied. Hence the final category weight is given as $\omega_i = sr_i \times f$ for each category.

5.3. Patch-based Attention Network

Additionally, we propose a novel patch-based attention network to deal with the imbalanced learning problem. The key idea is to force the network to learn a same number of kernel points for all categories by using an additional kernel loss. In our task, we have 6 categories in total and we select 8 kernel points per category. For each category, the ground truth kernel points are obtained by performing K-means algorithm on all points of this category. After 8 clusters are grouped, cluster centers are computed and their nearest neighbor points that belong to this category are defined as kernel points. The kernel loss is a L2 loss between the goal kernels and learned kernels. Hence the total loss in this case is defined as:

$$L_{\text{total}} = L_{\text{seg}} + \alpha L_{\text{rot}} + \beta L_{\text{ker}} \tag{2}$$

where β is a loss weight. We set $\beta = 0.05$. Apart from the obtained kernel points, their 32 neighbor points are grouped to form a patch and a convolution layer is used to get patch-wise features. The patch-wise features are further used to perform cross attention with the point-wise features learned from the DGCNN backbone (lower branch in Figure 6(a)). This is a patch-to-point cross attention, *i.e.*, using patch features to represent point features. The output is concatenated back to the lower branch for final segmentation. Detailed network designs are given in Figure 6.

5.4. Experimental results

All the experiments are conducted with the same dataset on which all augmentation methods are applied (dataset 4

in Table 2). Numerical results of them are presented in Table 3. From it, we can observe that the focused sampling strategy always leads to a worse performance. One possible reason is that this operation is only performed during training. For test cases, labels are segmentation goals and are not provided hence the focused sampling operation is not applicable. This causes data distribution difference between the training set and the test set thus leads to bad performance. The loss weighting strategy improves the performance on synthetic data but degrades the performance on real-world data slightly. This indicates the strategy contributes to a better pre-training yet not performing well on transfer learning. On the other hand, our proposed patch-based attention module improves the performance on both steps. To give better insights of our proposed module, learned patches of some sub-point clouds are visualized in Figure 7. It shows that our method forces the network to learn patches around boundaries or other informative places.

6. Conclusion

In this paper, we adopt sim2real transfer learning method for an industrial application on point cloud data. Following the pipeline, synthetic dataset are generated in simulated scenes. The network model is firstly pre-trained on the synthetic data and then fine-tuned on the real-world data. Both quantitative and qualitative results show that this achieves better performance. To deal with the imbalanced learning problem, several strategies have been tested. The proposed patch-based attention module shows its effectiveness by improving the performance drastically. For future directions, we would like to try with more backbones, as well as investigating more on the attention-based learning methods for point cloud data.

Acknowledgements

The project AgiProbot is funded by the Carl Zeiss Foundation.

References

[1] Juan Felipe Perez-Juste Abascal, Nicolas Ducros, Valeriya Pronina, Simon Rit, Pierre-Antoine Rodesch, Thomas Broussaud, Suzanne Bussod, Philippe Douek, Andreas Hauptmann, Simon Robert Arridge, and Françoise Peyrin. Material decomposition in spectral ct using deep learning: A sim2real transfer approach. *IEEE Access*, 9:25632–25647, 2021.

[2] Karol Arndt, Murtaza Hazara, Ali Ghadirzadeh, and Ville Kyrki. Meta reinforcement learning for sim-to-real domain adaptation. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2725–2731, 2020.

[3] Nicolas Audebert, Bertrand Le Saux, and Sébastien Lefèvre. Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. In *Asian conference on computer vision*, pages 180–196. Springer, 2016.

[4] Christian Bonatti, Sylvain Crovisier, Lorenzo Diaz, and Amie Wilkinson. What is... a blender? *arXiv preprint arXiv:1608.02848*, 2016.

[5] Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. Unstructured point cloud semantic labeling using deep segmentation networks. *3DOR@ Eurographics*, 3, 2017.

[6] Angel X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Qixing Huang, Zimo Li, S. Savarese, M. Savva, Shuran Song, Hao Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *ArXiv*, abs/1512.03012, 2015.

[7] Can Chen, Luca Zanotti Fragonara, and Antonios Tsourdos. Gapnet: Graph attention based point neural network for exploiting local feature of point cloud. *Neurocomputing*, 438:122–132, 2021.

[8] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019.

[9] Yuqing Du, Olivia Watkins, Trevor Darrell, P. Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1290–1296, 2021.

[10] Nico Engel, Vasileios Belagiannis, and Klaus C. J. Dietmayer. Point transformer. *IEEE Access*, 9:134826–134840, 2021.

[11] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996.

[12] David Griffiths and Jan Boehm. Synthcity: A large scale synthetic point cloud. *ArXiv*, abs/1907.04758, 2019.

[13] Michael Gschwandtner, R. Kwitt, A. Uhl, and W. Pree. Blensor: Blender sensor simulation toolbox. In *ISVC*, 2011.

[14] Meng-Hao Guo, Junxiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph Robert Martin, and Shimin Hu. Pct: Point cloud transformer. *Comput. Vis. Media*, 7:187–199, 2021.

[15] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Agathoniki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11105–11114, 2020.

[16] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018.

[17] Mingyang Jiang, Yiran Wu, Tianqi Zhao, Zelin Zhao, and Cewu Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018.

[18] Samin Khan, Buu Phan, Rick Salay, and Krzysztof Czarnecki. Procsy: Procedural synthetic dataset generation towards influence factor studies of semantic segmentation networks. In *CVPRW*, pages 88–96, 2019.

[19] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, A. Artemov, Evgeny Burnaev, M. Alexa, D. Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. *CVPR*, pages 9593–9603, 2019.

[20] Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep projective 3d semantic segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pages 95–107. Springer, 2017.

[21] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018.

[22] Le Hoang-An, Thomas Mensink, Partha Das, Sezer Karaoglu, and Theo Gevers. Eden: Multimodal synthetic dataset of enclosed garden scenes. *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1578–1588, 2021.

[23] Zhidong Liang, Ming Yang, Hao Li, and Chunxiang Wang. 3d instance embedding learning with a structure-aware loss function for point cloud segmentation. *IEEE Robotics and Automation Letters*, 5:4915–4922, 2020.

[24] Nai Qing Liu, Yinghao Cai, Tao Lu, Rui Wang, and Shuo Wang. Real–sim–real transfer for real-world robot control policy learning with deep reinforcement learning. *Applied Sciences*, 2020.

[25] Mohammadhossein Malmir, Josip Josifovski, Noah Klarmann, and Alois Knoll. Robust sim2real transfer by learning inverse dynamics of simulated systems. 2020.

[26] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[28] Kaichun Mo, Shilin Zhu, Angel Chang, Li Yi, Subarna Tripathi, Leonidas Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. 2019.

[29] Alexander Pashevich, Robin Strudel, Igor Kalevatykh, Ivan Laptev, and Cordelia Schmid. Learning to augment synthetic images for sim2real policy transfer. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2651–2657, 2019.

[30] R. Pierdicca, M. Mameli, E. Malinverni, M. Paolanti, and E. Frontoni. Automatic generation of point cloud synthetic dataset for historical building representation. In *AVR*, pages 203–219, 2019.

[31] C. Qi, Hao Su, Kaichun Mo, and L. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CVPR*, pages 77–85, 2017.

[32] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[33] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *ArXiv*, abs/1709.10087, 2018.

[34] Jacob Revell, Dominic Welch, and James M. Hereford. Sim2real: Issues in transferring autonomous driving model from simulation to real world. *SoutheastCon 2022*, pages 296–301, 2022.

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597, 2015.

[36] G. Ros, Laura Sellart, Joanna Materzynska, David Vázquez, and Antonio M. López. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. *CVPR*, pages 3234–3243, 2016.

[37] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.

[38] Philip Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The princeton shape benchmark. *Proceedings Shape Modeling Applications, 2004.*, pages 167–178, 2004.

[39] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017.

[40] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018.

[41] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019.

[42] Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[43] Jonathan Tremblay, Thang To, and Stan Birchfield. Falling things: A synthetic dataset for 3d object detection and pose estimation. *CVPRW*, pages 2119–21193, 2018.

[44] Juliano Vacaro, Guilherme Marques, Bruna Oliveira, Gabriel Paz, Thomas S. Paula, Wagston Staehler, and David Murphy. Sim-to-real in reinforcement learning for everyone. *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, pages 305–310, 2019.

[45] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. *CVPR*, pages 4627–4635, 2017.

[46] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018.

[47] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[48] Matthew Witman, Dogan Gidon, David B. Graves, Berend Smit, and Ali Mesbah. Sim-to-real transfer reinforcement learning for control of thermal effects of an atmospheric pressure plasma jet. *Plasma Sources Science and Technology*, 2019.

[49] Wenxuan Wu, Zhongang Qi, and Fuxin Li. Pointconv: Deep convolutional networks on 3d point clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9613–9622, 2019.

[50] Zhirong Wu, Shuran Song, A. Khosla, F. Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. *CVPR*, pages 1912–1920, 2015.

[51] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto L. Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2100–2110, 2019.

[52] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point transformer. *ArXiv*, abs/2012.09164, 2020.

[53] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, 2020.

[54] Qingnan Zhou and A. Jacobson. Thingi10k: A dataset of 10, 000 3d-printing models. *ArXiv*, abs/1605.04797, 2016.

[55] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109:43–76, 2021.