Supplementary Materials for “Quasi-Balanced Self-Training on Noise-Aware Synthesis of Object Point Clouds for Closing Domain Gap”

Yongwei Chen[1,2*], Zihao Wang[1*], Longkun Zou[1], Ke Chen[1,3†], and Kui Jia[1,3†]

[1] South China University of Technology
[2] DexForce Co. Ltd.
[3] Peng Cheng Laboratory
{eecyw,eezihaowang,eelongkunzou}@mail.scut.edu.cn,
{chenk,kuijia}@scut.edu.cn

A Details for Simulating Virtual Active Stereo Based Depth Sensor

In this section, we give a detailed description for the virtual active stereo based depth sensor simulation presented in Sec. 3 of our main text. The whole pipeline can be divided into building a physical simulator of stereo cameras for a photorealistic rendering (see Sec. A.1) and a pixel-to-point generation (see Sec. A.2).

A.1 Rendering with Projection of Speckle Patterns

For simulating a typical active stereo based depth sensor, we first set a stereo camera and a projector, which can actively project pre-defined speckle patterns in a simulation platform, e.g. the Blender [3] adopted in this paper. Specifically, the cycle engine in Blender [3] is employed as our rendering engine, which is a physically-based path tracer and supports artistic control and flexible shading nodes. Given object models $\mathcal{M} = \{M_i, y_i\}_{i=1}^{N}$ as input, the output of stereo image rendering can be $\{I_i^{\text{left}}, I_i^{\text{right}}, y_i\}_{i=1}^{N}$.

Pre-processing for Photorealistic Rendering – During the procedure of data pre-processing, those mesh-based models $\mathcal{M}$ are first scaled to a unit-ball with an arbitrary rotation along the z-axis, which are fed into the render engine together with the settings of scene illumination and object reflection for a photorealistic rendering. To this end, we place one area light source on top of the object model to be rendered, which can thus provide a uniform scene illumination. Moreover, characteristics of object material also affect the reflection of light received on object surface and are sensitive in optical imaging. In view of this, the bidirectional scattering distribution function (BSDF) material [2] is adopted in our generation to model the scattered pattern of light by a surface,

*Equal contribution
†Corresponding authors

by following the default setting of the BSDF function in [3] to initialize object models' material.

Virtual Configuration of A Stereo Camera – Given two identical optical imaging sensors (i.e. cameras O_L and O_R in Fig. 2 of our main text) for a stereo camera, the image planes of two cameras are coplanar. The left camera O_L is first placed near object model M with its optical axis towards the object, while the right camera O_R is set along parallel optical axes as O_L, with translation between two cameras only along the x-axis of the left camera. The simulated stereo camera fixes a baseline distance b between two cameras O_L and O_R, which are set as the simple pinhole camera using the same camera intrinsic parameters, e.g. focal length f, defined as prior knowledge in our simulation.

Projection of Extra Speckle Texture – Based on the theory about projection of stereo images [9], disparity of a point in 3D space in stereo images can be used to measure its depth distance, if correspondence of projected pixels on I^{left} and I^{right} can be discovered. However, due to texture-less appearance (e.g. CAD models in the ModelNet [11]), it could lead to mis-matching corresponding pixels in stereo images. A typical solution is to actively add pre-defined texture pattern on the object surface to provide visual appearance, which can thus benefit to discovering pixel correspondence in stereo matching. As a result, we use spot light in the cycle engine of Blender [3] to implement the projector P (refer to Fig. 2 of the main text), which is positioned in the middle of cameras O_L and O_R and along parallel optical axes of both cameras. Similar to real-world active stereo based depth sensors, the speckle image, in the form of a binary mask of white dots, is selected to provide an extra texture appearance on the object surface during generation. Technically, speckles are linked to the image texture node in spot light's nodes group, so that emission of light preserves the speckle pattern.

A.2 A Pixel-to-Point Generation

For obtaining a point cloud $\mathcal{P}$, depth images are first gained via stereo matching from the RGB images I^{left} and I^{right} (i.e. the output of stereo rendering in Sec. A.1), which can be further projected into 3D space using intrinsic parameters and extrinsic poses of the camera. To mimic generation of realistic point clouds from RGB-D videos scanning object in multiple poses in practice, synthetic point clouds under multiple camera poses are fused together to produce dense sets, which are then down-sampled to the final point set $\mathcal{P}$.

Stereo Matching – We perform stereo matching [9] on the output I^{left} and I^{right} of the image rendering to measure disparity between projection of one observed 3D point on I^{left} and I^{right}. $(u^{\text{left}}, v^{\text{left}}) \in I^{\text{left}}$ and $(u^{\text{right}}, v^{\text{right}}) \in I^{\text{right}}$ are defined as the 2D coordinates of the projected pixels on two image planes respectively. As the optical axes of two image planes in the stereo camera are set to be parallel, for each pixel in the left image, we only need to search along with the epipolar line to retrieval its corresponding one in the right image. Disparity d between a pair of matched pixels can be calculated by the following formula: $d = u^{\text{left}} - u^{\text{right}}$. In practice, we directly leverage block matching algorithm of

Algorithm 1: Quasi-Balanced Self-Training

```
Input     : Warm-up model $\Phi_0$, Init model $\Phi_{init}$, Target data $X_t$
Parameter: Constant $\epsilon$, Iteration I
Output    : Model $\Phi_I$
1  initialization $\theta_0$
2  for $i \leftarrow 1$ to $I$ do
3      $P_{index}, P_{value} = \arg\max \Phi_{i-1}(X_t), \max \Phi_{i-1}(X_t)$
4      $Mask = P_{index} > \theta_{i-1}$
5      $P_{index}, P_{value} = P_{index}[Mask], P_{value}[Mask]$
6      $X_t = X_t[Mask]$
7      $\widehat{Y}_t = onehot(P_{index})$
8      $L = sum(\widehat{Y}_t)$
9      for $k \leftarrow 1$ to $K$ do
10         $P^k_{value} = P_{value}[P_{index} = k]$
11         $X^k_t, \widehat{Y}^k_t = X_t[P_{index} = k], \widehat{Y}_t[P_{index} = k]$
12         $L_k = sum(\widehat{Y}^k_T)$
13         $\mu_k = 1 - L_k \ / \ L$
14         $L_k = ceil(L_k \times \mu_k)$
15         $sid = argsort(P^k_{value}, descending)$
16         $X^k_t, \widehat{Y}^k_t = X^k_t[sid][: L_k], \widehat{Y}^k_t[sid][: L_k]$
17     end
18     $\theta_i = \theta_{i-1} + \epsilon$
19     if i=1 then
20         $\Phi_i = update(\Phi_{init}|X_t, \widehat{Y}_t)$
21     else
22         $\Phi_i = update(\Phi_{i-1}|X_t, \widehat{Y}_t)$
23     end
24 end
25 return $\Phi_I$
```

the OpenCV to compute disparity. Its depth distance z, i.e. the distance between the 3D point and O_L along its optical axis, can be computed with focal length f and baseline b of the camera as follows [9]: $z = (f \cdot b)/d$.

Depth Transformation – 2D coordinate (u, v) and depth value z of every pixel in the depth image is transformed to a point (x, y, z) in 3D space as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = z \begin{bmatrix} 1/f & 0 & 0 \\ 0 & 1/f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u - c_x \\ v - c_y \\ 1 \end{bmatrix} \quad (1)$$

where (c_x, c_y) denotes the principal point. Note that the point cloud $\mathcal{P}$ calculated by Eqn. (1) lies in the camera coordinate system, which demands spatial transformation to the world coordinate system by using the camera pose consisting of a translation vector and a rotation matrix.

Multi-view Fusion of Point Clouds – As the same object can be observed by the RGB-D camera at different poses in real-world scanning, realistic point clouds thus carry richer geometric information under multiple observation angles. In light of this, multi-view synthetic point clouds are produced by following the aforementioned modules, which are then fused into a dense point cloud. For a fair comparison across different sizes of view, we then downsample the dense point cloud into a sparse one as the final output $\mathcal{P}$ of our generation method.

B Algorithm of Quasi-Balanced Self-Training

Algorithm 1 shows the details of the quasi-balanced self-training (cf. Sec. 3.2 in the main text).

C More Comparative Evaluation

Table 1. Classification accuracy (%) averaged over 3 seeds (±SEM) on the Real2Sim tasks.

Method	D→M	D→S
Pointnet++ [7]	58.4 ± 0.8	73.3 ± 0.9
DGCNN [10]	65.3 ± 1.3	76.7 ± 0.3
RSCNN [6]	63.4 ± 2.8	76.4 ± 0.2
SimpleView [5]	69.4 ± 1.3	74.4 ± 0.5

Real2Sim Cross-Validation – To further verify that the proposed Mesh-to-Point (Mesh2Point) pipeline can physically simulate the non-uniform noises of real point clouds to mitigate domain gap between synthetic and real point clouds, we conduct a complementary cross-validation experiment in a Reality-to-Simulation (Real2Sim) manner to complement Table 1 of the main text, with four representative point classifiers. In details, all the models are trained on the real DepthScanNet10 (**D**), while testing on the testing split of synthetic data (the proposed SpeckleNet10 (**S**) vs. the ModelNet10 (**M**) [8]). Superior performance in Table 1, i.e. higher accuracies in the **D→S** column than those in the **D→M** column, can demonstrate again the effectiveness of our Mesh2Point to generate synthetic data approaching statistical distribution of real data owing to physical simulation of systematic noises in stereo rendering of projected speckle patterns and matching.

Evaluation on Original ScanNet – We report comparison between the proposed QS^3 scheme and other UDA methods on the less challenging ScanNet ($\mathbf{S}^*$) [4], using the same UDA experiment setting. Results in Table 2 again demonstrate the superiority of our method to other competiong methods.

Table 2. Evaluation in classification accuracy (%) of point classifiers training on the SpeckleNet and testing on the ScanNet

Method	S→S*
Supervised	78.7 ± 0.8
DGCNN [10] (w/o Adapt)	51.1 ± 1.2
PointDAN [8]	53.5 ± 0.8
DefRec [1]	50.9 ± 0.1
DefRec+PCM [1]	56.1 ± 0.2
GAST w/o SPST [12]	49.3 ± 1.1
GAST [12]	51.9 ± 0.9
QS^3 (ours)	**57.4 ± 0.2**

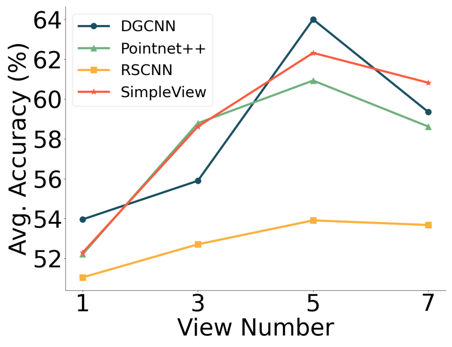

Fig. 1. Effects on varying number of views in multi-view fusion of synthetic point clouds on the **S→D** task.

D More Ablation Studies

Effects on Multi-View Fusion – We also evaluate effects of varying sizes of single-view synthetic point clouds to be fused on model generalization to real data, which is visualized in Fig. 1. Generally speaking, the more views, the higher the classification accuracy. Moreover, fusion with single-view point clouds under five camera poses is preferred owing to its consistently better performance with all four methods.

Effects of θ_0 and ϵ in the Quasi-Balanced Self-Training – Ablation studies of our QBST are investigated on two hyper-parameters, i.e. the initial threshold θ_0 and ϵ in Algorithm 1 of the main text, which can be sensitive to UDA classification performance. Specifically, θ_0 on filtering prediction confidence is used to control the number of selected pseudo-labeled samples for self-training, while ϵ is the constant step to gradually increase θ during self-training iterations. Fig. 2 illustrates effects of varying θ_0 and ϵ on the **S→D** and **M→D** tasks. Generally, regardless of selection of θ_0 and ϵ, models training on our synthetic data from the SpeckleNet10 can consistently outperform those training on the ModelNet10 [8] with a significant margin, which can favor our motivation about the proposed

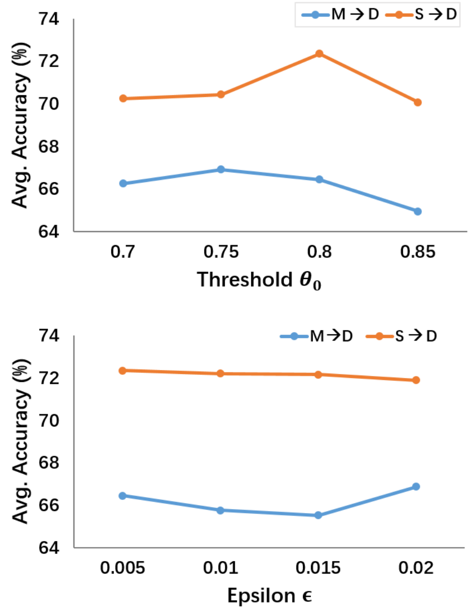

Fig. 2. Left: Effects of θ_0 on the **S→D** and **M→D** tasks; Right: Effects of ϵ on the **S→D** and **M→D** tasks in the case of $\theta_0 = 0.8$.

QS^3 scheme on closing Sim2Real domain gap. Moreover, stable performance of different values of θ_0 and ϵ can be achieved, owing to self-training with more balanced data distribution. As a result, in our QBST, the optimal value of θ_0 is set as 0.8, while $\epsilon = 0.005$.

E Visualization

We visualize point cloud examples from the generated SpeckleNet10 dataset using our Mesh2Point and the DepthScanNet10 adapted from the real ScanNet [4] in Figs. 3 and 4 respectively. Evidently, visual similarity of examples from both datasets can confirm our motivation of physical simulation of real point clouds.

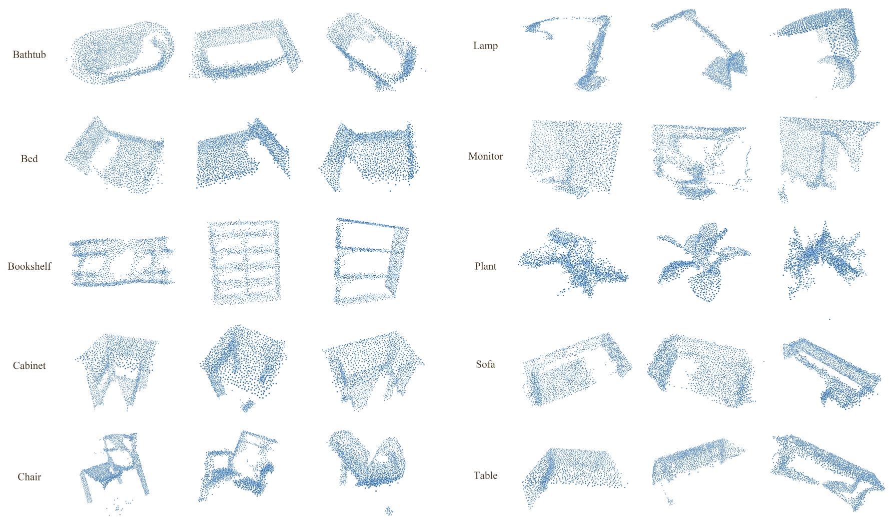

Fig. 3. Visualization of synthetic examples from the generated SpeckleNet10.

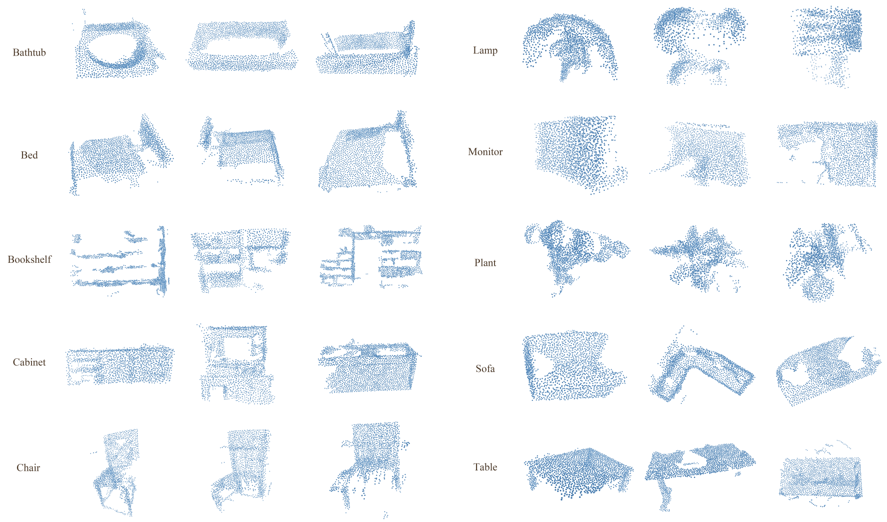

Fig. 4. Visualization of real examples from the DepthScanNet.

References

1. Achituve, I., Maron, H., Chechik, G.: Self-supervised learning for domain adaptation on point clouds. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 123–133 (2021) 5
2. Bartell, F.O., Dereniak, E.L., Wolfe, W.L.: The theory and measurement of bidirectional reflectance distribution function (brdf) and bidirectional transmittance distribution function (btdf). In: Radiation Scattering in Optical Systems (RSOS). vol. 257, pp. 154–160. SPIE (1981) 1
3. Community, B.O.: Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam (2018), `http://www.blender.org` 1, 2
4. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5828–5839 (2017) 4, 6
5. Goyal, A., Law, H., Liu, B., Newell, A., Deng, J.: Revisiting point cloud shape classification with a simple and effective baseline. In: International Conference on Machine Learning (ICML). pp. 3809–3820. PMLR (2021) 4
6. Liu, Y., Fan, B., Xiang, S., Pan, C.: Relation-shape convolutional neural network for point cloud analysis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8895–8904 (2019) 4
7. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in Neural Information Processing Systems (NeurIPS) **30** (2017) 4
8. Qin, C., You, H., Wang, L., Kuo, C.C.J., Fu, Y.: Pointdan: A multi-scale 3d domain adaption network for point cloud representation. Advances in Neural Information Processing Systems (NeurIPS) **32** (2019) 4, 5
9. Szeliski, R.: Computer vision: algorithms and applications. Springer Science & Business Media (SSBM) (2010) 2, 3
10. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG) **38**(5), 1–12 (2019) 4, 5
11. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1912–1920 (2015) 2
12. Zou, L., Tang, H., Chen, K., Jia, K.: Geometry-aware self-training for unsupervised domain adaptation on object point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 6403–6412 (2021) 5