Object Instance Matching from Partial Data

Chuqiao Li
EEIT ETHz
Switerland
chuqli@student.ethz.ch

Yiming Zhao
EEIT ETHz
Switerland
yimzhao@student.ethz.ch

Abstract: When building a map of the environment robots oftentimes have to rely on partial observations due to occlusions and limited field of view. Simultaneously, deep networks have shown the capability to learn strong object shape priors to generalize from partial data. Based on [1], we show that learned shape representation latent space can be leveraged in 3D object re-localization process. Our adapted model has improved the partial-to-partial matching score and at the same time achieving better reconstruction results.

Keywords: 3D partial Reconstruction, Matching, Learning, Latent Space, Signed Distance Function

1 Introduction

Visual perception of the surrounding world is very important to robots whether in the tasks of localizing itself or interacting with other objects. To establish an accurate perception of the environment, 3D object alignment is essential. And as the development of sensors, RGB-D images can be easily acquired by robots. For 3D object detection and localization, there are various methods. For instance, some methods based on local descriptors would require the data to contain sufficient amount of texture, e.g.[2]. And in the extreme cases where depth images are from two viewpoints that have no overlap, it is problematic for key-point matching. This can be solved by applying template-based approaches, but the rapid growth of templates stored in memory will be needed with more instances. Similar to the idea in [3] of encoding pose information in templates and then compare them with input data, we propose to encode learned 3D object shape information in representation space.

Whether the task is tracking target, object detection, or instance re-localization, a previous matching step is always needed. With the depth-images partial data from robots, our goal to reconstruct the instances and also match objects from partial data. Based on DeepSDF[1], we purpose an adapted model which can derive a 3D shape representation space that could be applied for matching 3D objects rendered by single viewpoints. Our model is expected to improve matching between different partial observations of the same objects, generate close SDF values for those observations also output better reconstruction from partial inputs.

Our contributions compare to the original model is as following:

- We leverage generative models to do shape completion for instance re-localization.
- We make latent space more meaningful for partial-to-partial matching and alignment by implementing various latent space losses.
- Improvement on completion process with lower reconstruction loss by adapting partial data as training inputs.
- Improvement on training speed by using distributed computing and Apex mixed-precision optimizer.

2 Related Work

We review two main areas of related work: 3D object detection and localization and 3D shape representation.

5th Conference on Robot Learning (CoRL 2021), London, UK.

3D object detection and localization There is a long history and thus fruitful research results in the area of 3D object detection and localization. The researches are based on photo-metric images and range images, and more recently, registered color/depth images. Among them, RGB-D images can be easily acquired and also efficiently used for the detection and localization process. In the following brief review, we focus on techniques based on techniques that solve the detection and localization of objects in realistic situations, such as cluttered or changing scenes. The researches in this area can be roughly classified into three categories: template-based approaches, dense approaches, feature-based approaches.

Template-based approaches are one of the traditional methods for instance detection, and they scan through the whole image to match some rigid templates, e.g.[4]. For instance, in [3], templates that contain different viewpoints of 3D shape texture-less objects are generated by RGB-D sensors. But they also have some limitations such as requiring known templates, which could be a potential issue when new instances emerge.

Dense-based approach leverage voting method, every pixel gives a prediction of the desired output. For example, [5] leverages a learned, intermediate representation in form of a dense 3D object coordinate labeling paired with a dense class labeling. But these methods can be affected by the quality of images, e.g. very low resolution.

Feature-based approaches could depend on local descriptors, such as [6] which demonstrate the experiment on 6D-Pose Estimation and Re-localization of objects in real scenes based on key-point matching. However, in the situation where two viewpoints of the same instance have little overlapping, then matching key points are problematic. Global descriptors can also be helpful. In another novel paper, Scene Representation Networks(SRNs)[7] could learn from the posed 2D images to generate scene representations. And the author provided an insight into the latent vector. However, the SRNs require camera intrinsic and extrinsic parameters which are not always available in real cases. In our work, instead of encoding scene knowledge, we encoded prior shape knowledge from massive data of the same class in learned 3D shape representation. The information about the object encoded in the learned representation can be used in the progress of localization.

3D shape representation The literature of 3D shape representation can be classified into three categories: point-based, mesh-based, and voxel-based methods. Point-based methods, e.g.[8, 9], require sampling points from the surface of objects which could achieve from the raw output of many sensors, such as depth cameras. However, point clouds do not have topology and thus not helpful for generating watertight surfaces. Mesh-based methods leverage the predefined template meshes which represent classes with similar shapes, e.g.[10, 11]. [12] propose to generate valid meshes by progressively deforming from the initial template shapes. However, a potential problem of this method is local coherence. Voxel-based method use 3D grids of values to represent 3D volume, one example is occupancy grid[13] which is currently limited to low resolution. Another way is to use 3D grids of values that represent signed distance functions(SDF), e.g.[14], which is memory-consuming by using discrete voxels.

The paper ScanComplete[15] introduced an approach of using fully-convolutional neural networks to transform incomplete SDFs into complete meshes at unprecedented spatial extents. This method starts also from the SDF and its data pre-processing is a good reference for our setup. But the main difference to our purpose is that ScanComplete is applied for relatively large scene completion but not focuses on the Re-Localization of a single object. Our work is based on DeepSDF[1] which leverages SDF input and has shown good performance on shape representation and completion. But the problem is there has no explore in the representation space (latent space) of single viewpoints and whether it can be used for 3D object matching and localization.

In our work, we explore the shape representation space (latent space) of different single-viewpoints and make adaptations to[1] to improve the matching score in latent space.

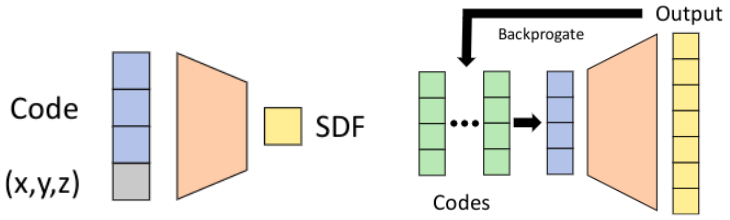
3 Method

After exploring latent space, we found that an implicit relation exists between latent codes of different partial observations of the same instance. Considered that the latent space is nonlinear, the simple linear interpretation could not represent this relationship in most cases. And due to arbitrary initialization, the structure and order of elements in latent code could be different for each train-

ing. Therefore, our basic idea is to enable the network in training time to learn the similarity of partial data of the same object and even distinguish the difference between another object. Under this assumption, the inferred latent codes of different partials of the same object should have smaller distances in latent space to each other than the codes from other objects. According to the nearest neighbor of the latent code, it could be determined which partial observations belong to the same object. This information might be very helpful for instance re-localization.

3.1 Model Adaptation

Our method is based on DeepSDF model[1] which has the architecture of auto-decoder. During training time, it takes a single point(x,y,z) and a latent vector as input and predicts the corresponding SDF values of this point. After calculating the l1 loss between true SDFs and predicted SDFs of the input point clouds, it does backward-propagation on the weights of the auto-decoder and also the latent vector. And during the test procedure, it takes in partial noisy test data and optimizes latent code based on predicted SDF values of test data points. Conditioning on the optimized latent vector, we query the decoder for other points of the whole space to get the reconstructed complete shape.



(a) Coded Shape DeepSDF (b) Auto-decoder

Figure 1: DeepSDF model architecture

Since the original model takes a long time for training a single epoch, we developed a distributed architecture with APEX mixed-precision optimizer, which could drastically accelerate the training and meanwhile keep the performance in the same magnitude. The improvement could be seen in the following table. This architecture could be helpful to test new models under complex networks and a large amount of data.

Model	avg. Computation Time per Epoch/s	avg. Test Error	Error Variance
Original Model	160s	0.00375	0.00122
With Adaptation	18s	0.00400	0.00163

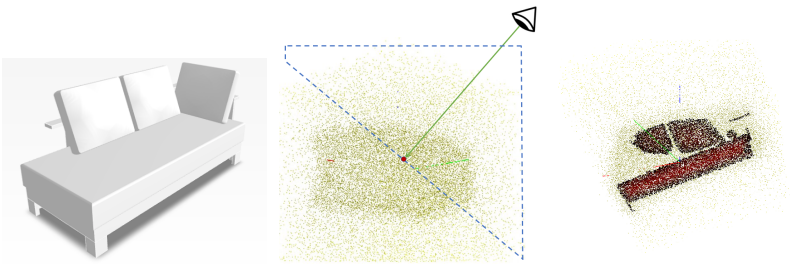
Table 1: Training speed and loss comparison with 'sofa' dataset from ShapeNetCore.v2 with 1628 samples.

3.2 Data preparation

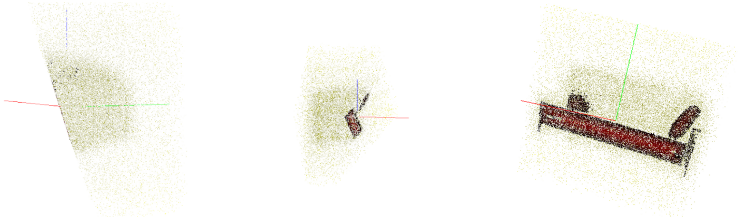
3.2.1 Train Data

In the previous experiments, we tried to train the network with data generated by depth image, which only samples the SDF value near the surface points. Then, the network has no ability to precept the distribution of SDF value in the whole space. Hence, the performance of the model was not ideal. Thus, we apply a virtual cut to the obtained SDF data, which are sampled by the whole mesh and distributed in the whole space. The virtual cut could be implemented in the following steps. Firstly, a camera position is randomly selected on the sphere whose center is homocentric to the center of mass C of the observed object, Secondly, the virtual camera takes the look at the center and the camera ray also is defined from the center to the camera position. Thirdly, we introduce a cut plane with normal n along the camera ray and through the center. Then all the SDF samples in the half-space $\vec{\mathbf{n}}(\mathbf{x} - C) > 0$ will be kept. The virtual cut could approximate the distribution of

SDF samples on the observed surface by depth image and also have the spatial information in the surrounding space.



(a) Mesh (b) SDF Sample in Space with Cut (c) Cut 1



(d) Cut 2 (e) Cut 3 (f) Cut 4

Figure 2: Preparation of Train Data

3.2.2 Test Data

In comparison to the train data, the test data is generated by the depth image. We use Blender to obtain the depth image which is rendered with the resolution 380x250. We reproject the pixels into space to get the corresponding point cloud for the observed partial surface. For each point, we defined the point normal along the direction from this point to camera position. With a specified distance ϵ, the SDF value could be sampled alone the point normal near the surface for both sides.

(a) Mesh (b) Depth Image (c) SDF near Surface

Figure 3: Preparation of Test Data

3.3 Distributed Training with constraint loss

Instead of direct feeding the partial data into the network, we develop a new training procedure "Batch in Batch" based on a distributed architecture. We divided the GPUs into multiple groups. The different partial data of the same instance will be trained within the group. Then besides the original L1 loss, an additional constraint loss cross GPUs is introduced. We develop three kinds of loss and in the backward propagation of those losses, only the gradients of latent codes will be computed, More specifically, those losses are used to optimize the latent codes.

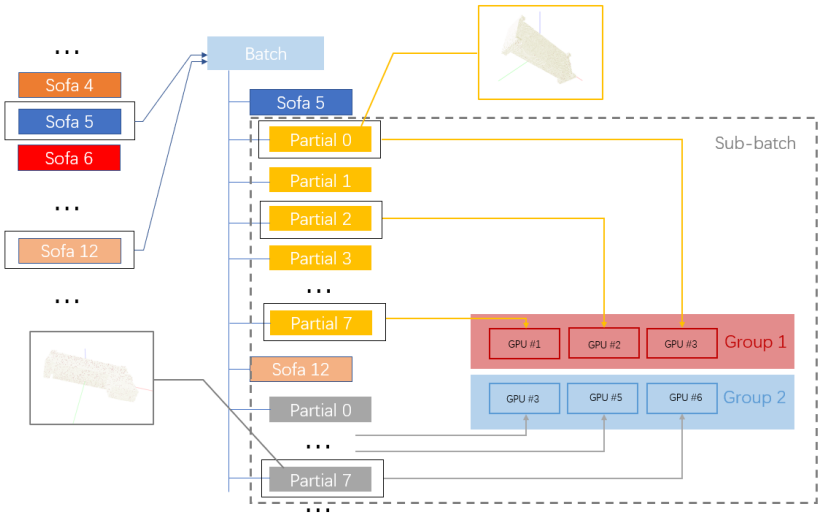


Figure 4: Example for **"Batch in Batch"** with 6 Threads on 6 GPUs in 2 Groups

3.3.1 Constraint Loss

Mahalanobis Distance Loss we want to decrease the Mahalanobis distance between two latent vectors from the same instance. The covariance matrix $\mathbf{M}$ of Mahalanobis distance is computed by all latent vectors $\mathbf{Z}$. We denote the set $S(z_i)$ as all latent codes except $\mathbf{z_i}$ of partial data from the same instance. As shown in the following figure, on each thread, here we assign one thread per GPU, the latent vectors $\mathbf{z_j}$ in $S(z_i)$ will be broadcasted to the current thread. The loss takes the mean of Mahalanobis distance between latent vector of local thread to other broadcasted latent vectors. The optimizer of the local thread will take this loss to optimize the local latent vector$\mathbf{z_i}$. The above steps are parallelized on all GPUs. At the end of every epoch, the covariance matrix will be updated with a delay. This loss requires at least two parallel threads within a group.

$$\min_{z_i} \frac{1}{|S(z_i)|} \sum_{z_j \in S(z_i)} (z_i - z_j)^T \mathbf{M}^{-1} (z_i - z_j) \tag{1}$$

$$\mathbf{M}^{(0)} = \mathbf{I}_{d \times d}, \;\; \mathbf{M}^{(k+1)} = (1-\delta)\mathbf{M}^{(k)} + \delta \mathbf{cov}(\mathbf{Z}^{(k)}) \tag{2}$$

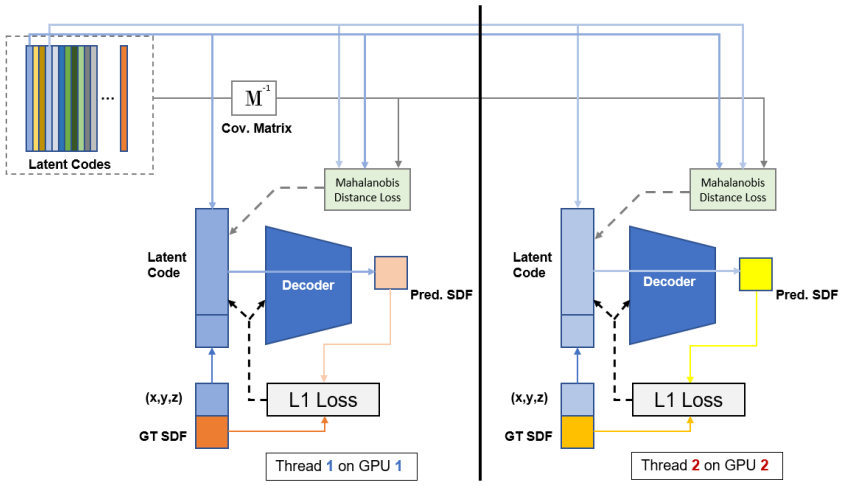


Figure 5: Explanation for **Mahalanobis Distance Loss** with 2 Threads on 2 GPUs

Group Latent Loss Group latent loss: Intuitively, The purpose of adding this constraint loss to encourage the decoder to output the same SDF value on queried point with different latent vectors but from the same instance. Similar to Mahalanobis distance loss, the local thread received the latent vectors from other threads. And those latent vectors, which belong to different partial data of the same instance, will be concatenated with local queried points xyz_i from the partial data on the local thread and inputted to the decoder $\mathbf{d}$. The loss is computed by the mean squared error between predicted SDF value with local latent vector to others. The local latent vector $\mathbf{z_i}$ will be optimized. Because the training is dynamic for the decoder and latent vectors. Then, the optimization of the latent vector also affects the decoder's optimization.

$$\min_{z_i} \frac{1}{|S(z_i)|} \sum_{z_j \in S(z_i)} (\mathbf{d}(z_i, xyz_i) - \mathbf{d}(z_j, xyz_i))^2 \tag{3}$$

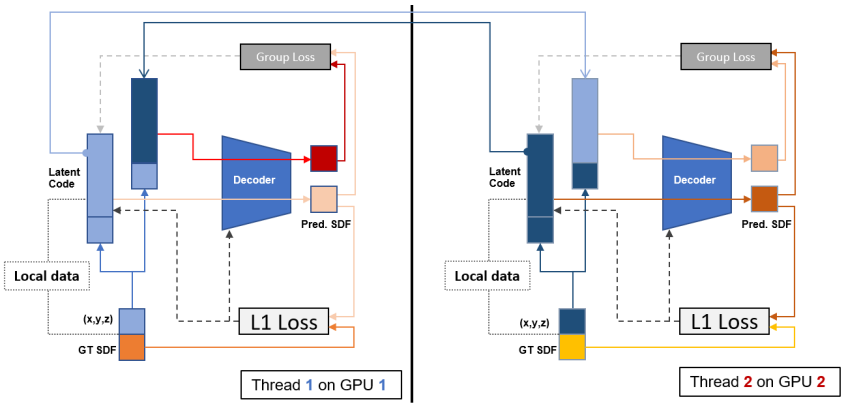


Figure 6: Explanation for **Group Latent Loss** with 2 Threads on 2 GPUs

Triplet-like Loss This loss is inspired by the triplet loss[16]. The implementation is similar to the Group latent loss. Its implementation requires at least two groups and each group with two GPUs. The decoder predicts additional SDF values for the latent vectors from different instances(denoted as $O(z_i)$) . We computed the mean squared error between them and the prediction of the local latent vector and add this error as a negative term to the Group Latent Loss, meanwhile, also add a positive margin. Considering the SDF values could also have small differences for the same sampled points around different instances, thus the negative term will be multiplied by a coefficient α to reduce its dominance. And so far the margin is fixed during training, it will be changed to be proportional to the L1 loss in the further experiment.

$$\min_{z_i} \max(\frac{1}{|S(z_i)|} \sum_{z_j \in S(z_i)} (\mathbf{d}(z_i, xyz_i) - \mathbf{d}(z_j, xyz_i))^2 - \frac{\alpha}{|O(z_i)|} \sum_{z_l \in O(z_i)} (\mathbf{d}(z_i, xyz_i) - \mathbf{d}(z_l, xyz_i))^2 + margin, 0) \tag{4}$$

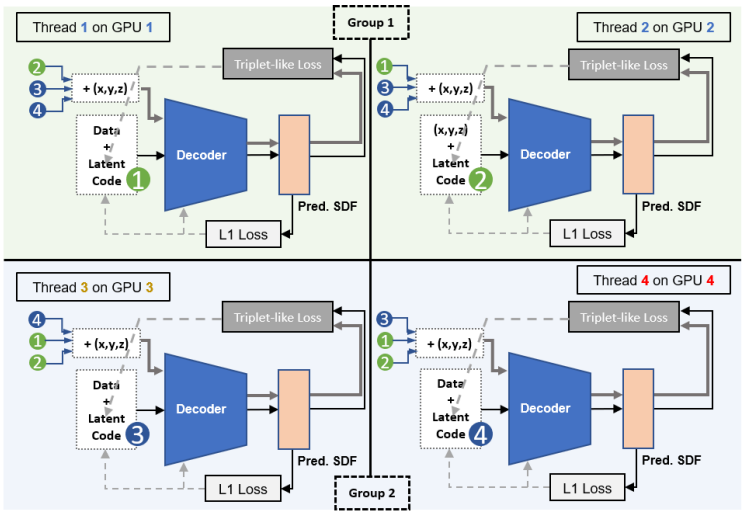


Figure 7: Explanation for **Triplet-like Loss** with 4 Threads on 4 GPUs

3.3.2 Weighted Loss

We believe that latent vectors of partial data from the same instances should not be the same but have a closer distance in latent space in some ways. The Optimization on L1-loss determines the overall performance of the decoder. The following constraint losses are regarded as an auxiliary method, we also adjust coefficient α to reduce its dominance. In the current experiment, the coefficient is fixed. It could be modified to a dynamic value during training, for example, proportional to the L1-loss.

$$\textbf{Loss} = \textbf{L1-loss} + \alpha\textbf{Constraint-loss} \tag{5}$$

3.3.3 Latent Distance in Evaluation

Instead of Euclidean distance, we also use the Mahalanobis distance to determine the nearest neighbor of each latent code in the evaluation stage, the covariance matrix is computed by the latent codes from the training stage at the last epoch.

3.3.4 Constraint Loss in Reconstruction

When we have already determined that some partial observations belong to the same instance, we could also add a proper constraint loss from above to their latent codes during the optimization. The reconstructed mesh has comparatively higher accuracy. The result of this experiment will be shown in the next section 4.4.

4 Experimental Results

All the training experiments were conducted on a remote computation server with 6 **RTX-2080 Ti** graphics card. The training experiments of **Mahalanobis Distance Loss** and **Group Latent Loss** were on one GPU group with 6 GPUs. The training experiment of **Triplet-like Loss** was on two GPU groups and each group with 3 GPUs.

Our train datas are **Sofa** test: 1628 instances with each instance 16 viewpoints; **Lamp** train: 897 instances with each instance 32 viewpoints; **Plane** train: 1780 instances with each instance 16 viewpoints.

We evaluate our model on three classes: sofa, lamp, planes from the dataset "ShapeNetCore"[17]. For each class, We compare our adapted models with three constraints losses and the original model on three aspects:

- Matching score in latent space
- Reconstruction visualization

- Reconstruction error

Our test datas are **Sofa** test: 411 instances with each instance 16 viewpoints; **Lamp** test: 213 instances with each instance 6 viewpoints; **Plane** test: 456 instances with each instance 6 viewpoints.

4.1 Training Loss

The following loss logs show the the change of loss during the training on class sofas. The learning rate is adjusted to halve for every 500 epochs. We could see that the curve of Mahalanobis Distance Loss decreases most smoothly, because the latent codes are directly optimized by Mahalanobis Distance. The other two losses are computed by the SDF prediction from the decoder. the Group Latent Loss increases dramatically at the very beginning due to random initialization of decoder and latent codes. And then, both of two losses descend with oscillation.

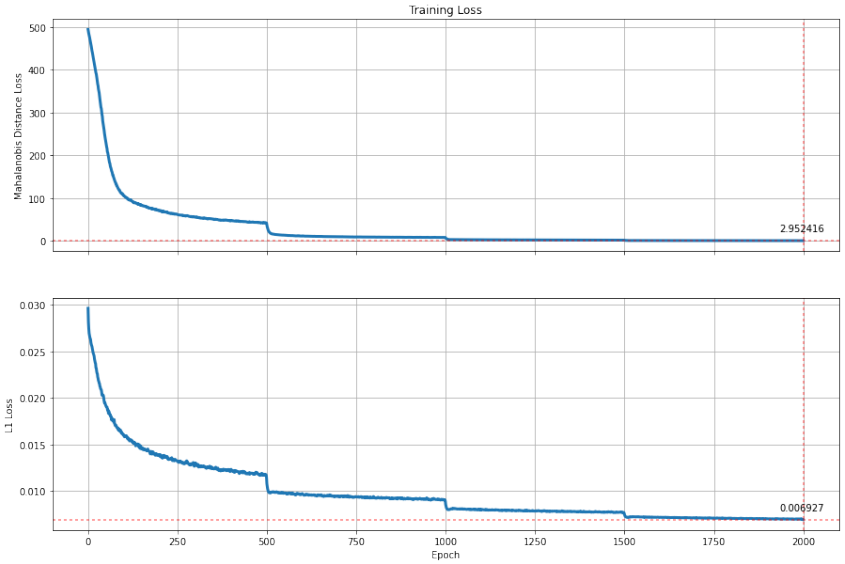


Figure 8: Training Loss Log of **Mahalanobis Distance Loss**

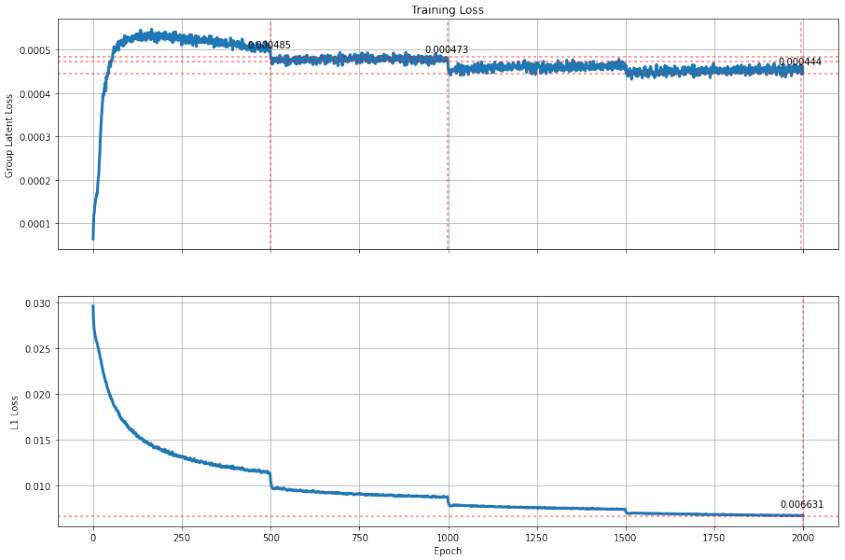


Figure 9: Training Loss Log of **Group Latent Loss**

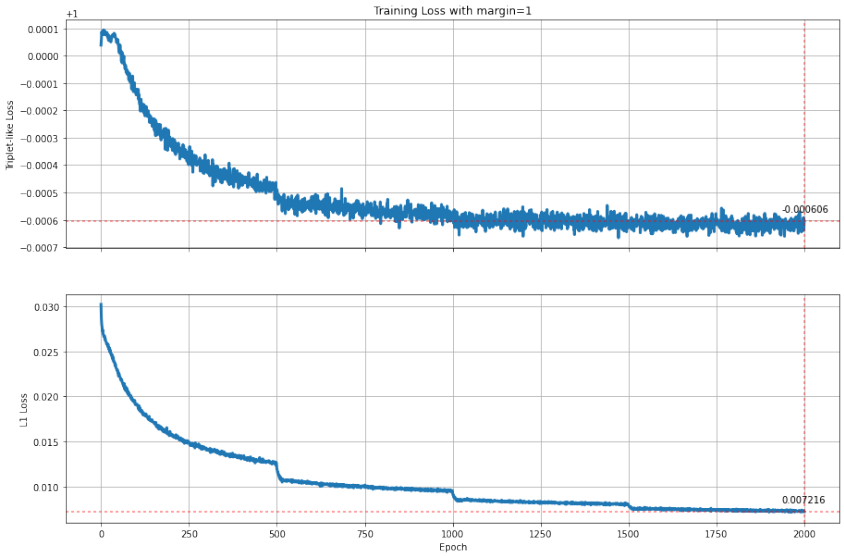


Figure 10: Training Loss Log of **Triplet-like Loss**

4.2 Matching Score in Latent Space

To evaluate partial-to-partial object matching accuracy in the latent space, we use **Top K Score**:

$$\textbf{Top k score} = \frac{1}{N}\sum_{x=1}^{N}\mathbf{I}(x) \tag{6}$$

where I is the indicator function, i.e. **I**(x) =1 if any of the k neighbors is correct, 0 otherwise. Higher score represents better performance.

4.2.1 Sofa

As shown in the following tables, we have improved the matching accuracy by adapt the model with the three implemented losses. Top k score in the latent space has been improved by using Mahalanobis distance instead of Euler distance. And for k=1, we have increased the best matching score(0.6321) to almost twice the original(0.3227). And as k increases, the score also increases. Scores when k =10 are almost twice when k = 1.

Top k Score(k = 1)	original	loss_md	loss_gl	loss_tri
Euler distance	0.3227	0.4227	0.4237	**0.4970**
Mahalanobis distance	0.4840	0.6141	0.5827	**0.6321**

Table 2: Top k score when k = 1, evaluated from original model and the models with three constraints: Mahalanobis Distance loss (Loss_md), Group Latent Loss (Loss_gl), Triplet-like Loss (Loss_tri)

Top k Score(k = 5)	original	loss_md	loss_gl	loss_tri
Euler distance	0.5409	0.6857	0.6624	**0.7346**
Mahalanobis distance	0.7401	0.8400	0.8251	**0.8514**

Table 3: Top k score when k = 5, evaluated from original model and the models with three constraints: Mahalanobis Distance loss (Loss_md), Group Latent Loss (Loss_gl), Triplet-like Loss (Loss_tri)

Top k Score(k = 10)	original	loss_md	loss_gl	loss_tri
Euler distance	0.6384	0.7836	0.7693	**0.8269**
Mahalanobis distance	0.8329	0.9056	0.8945	**0.9077**

Table 4: Top k score when k = 10, evaluated from original model and the models with three constraints: Mahalanobis Distance loss (Loss_md), Group Latent Loss (Loss_gl), Triplet-like Loss (Loss_tri)

4.2.2 lamp

As shown in the following tables, for the class lamp, our adapted models also outperform the original model in the latent space for matching.

Top k Score(k = 1)	original	loss_md	loss_gl	loss_tri
Euler distance	0.5798	0.6346	0.6299	**0.6674**
Mahalanobis distance	0.6307	0.7277	0.7097	**0.7340**

Table 5: Top k score with k = 1

Top k Score(k = 3)	original	loss_md	loss_gl	loss_tri
Euler distance	0.7535	0.8122	0.8255	**0.8333**
Mahalanobis distance	0.7731	**0.8803**	0.8607	0.8560

Table 6: Top k score with k = 3

Top k Score(k = 6)	original	loss_md	loss_gl	loss_tri
Euler distance	0.8388	0.8983	0.8983	**0.9077**
Mahalanobis distance	0.8482	**0.9327**	0.9163	0.9171

Table 7: Top k score with k = 6

4.2.3 planes

As shown in the following tables, for the class plane, our adapted models also outperform the original model in the latent space for matching. Two additional experiments were done for **Group Latent Loss** and **Triplet-like Loss**: we set α to 1 in (5). It means that the dominance of those losses will not be reduced. The results of above two experiments will be denoted as $loss\_gl\_nr$ and $loss\_tri\_nr$ in following tables.

Top k Score(k = 1)	original	loss_md	loss_gl	loss_gl_nr*	loss_tri	loss_tri_nr*
Euler distance	0.4926	0.5500	0.4682	0.5559	0.5581	**0.5673**
Mahalanobis distance	0.5690	**0.6338**	0.5154	0.6086	0.6176	0.6224

Table 8: Top k score with k = 1

Top k Score(k = 3)	original	loss_md	loss_gl	loss_gl_nr*	loss_tri	loss_tri_nr*
Euler distance	0.6293	0.7065	0.6005	0.6915	0.7032	**0.7292**
Mahalanobis distance	0.7028	**0.7825**	0.6586	0.7613	0.7650	0.7763

Table 9: Top k score with k = 3

Top k Score(k = 6)	original	loss_md	loss_gl	loss_gl_nr*	loss_tri	loss_tri_nr*
Euler distance	0.7105	0.7982	0.6769	0.7800	0.7978	**0.8216**
Mahalanobis distance	0.7202	0.8665	0.7452	0.8487	0.8501	**0.8673**

Table 10: Top k score with k = 6

4.3 Reconstruction visualization

We believe after we apply constraint loss for training model, the latent code, which is predicted for the partial SDF samples in test phase, could be directly used for mesh reconstruction without any further optimization. Generally, the performance of this reconstruction is decided by the position of viewpoint and whether the test sample could be generally represented by the train sample. In comparison to lamp class, the shape and structure of instances in sofa class and in plane class are relatively more regular. However, the reconstruction could also locally fail on details for some "difficult" sample.

4.3.1 Sofa

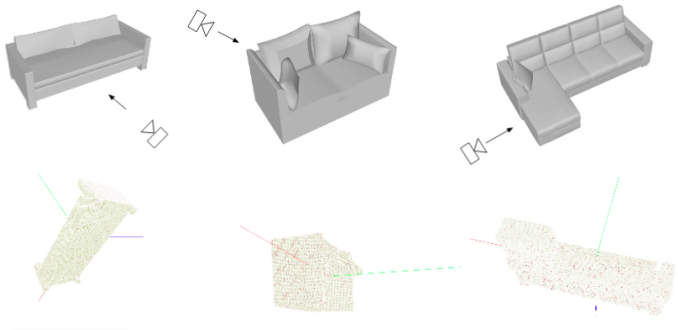
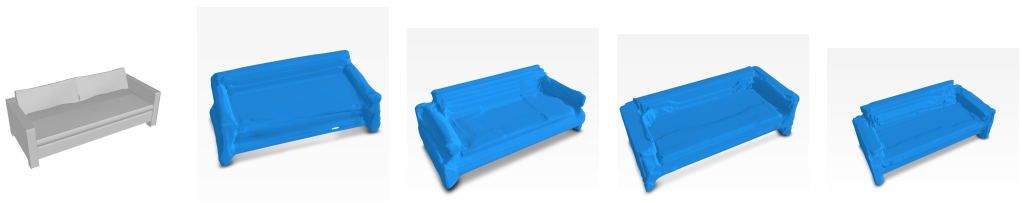
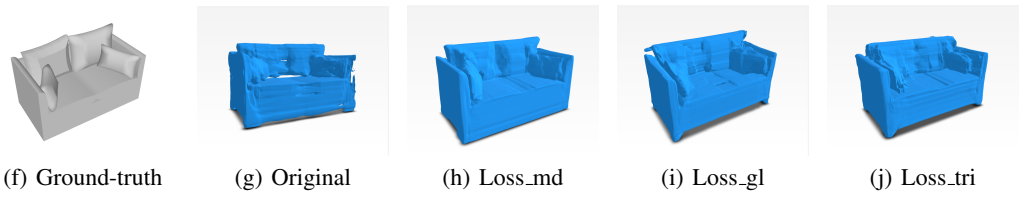


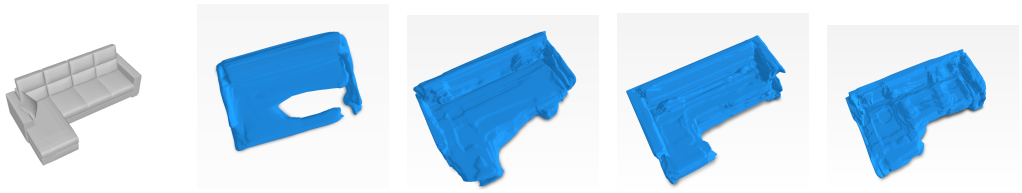
Figure 11: Test input partial SDFs the three instaces



(a) Ground-truth (b) Original (c) Loss_md (d) Loss_gl (e) Loss_tri



(f) Ground-truth (g) Original (h) Loss_md (i) Loss_gl (j) Loss_tri



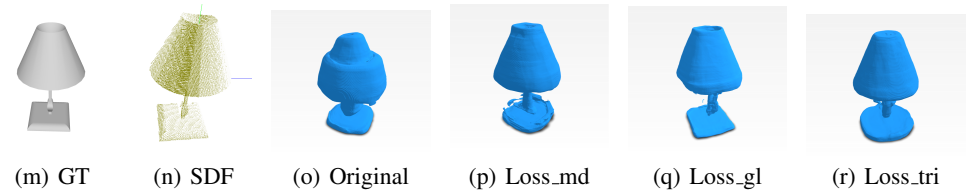
(k) Ground-truth (l) Original (m) Loss_md (n) Loss_gl (o) Loss_tri

Figure 12: Output reconstructed meshes of three instances from original model and the models with three constraints: Mahalanobis Distance loss ($Loss\_md$), Group Latent Loss ($Loss\_gl$), Triplet-like Loss ($Loss\_tri$)

4.3.2 Lamp

(a) GT (b) SDF (c) Original (d) Loss_md (e) Loss_gl (f) Loss_tri

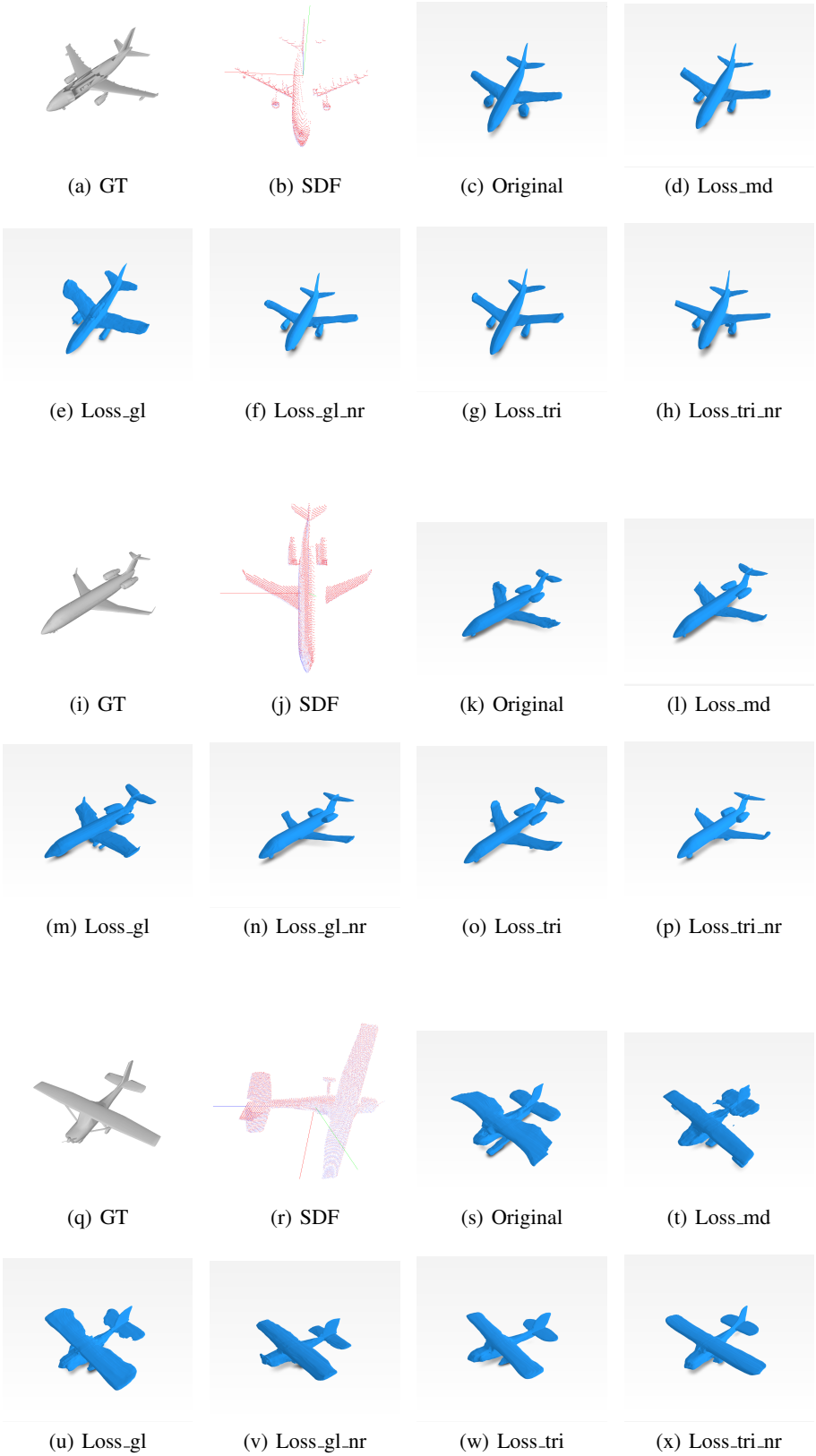
(g) GT (h) SDF (i) Original (j) Loss_md (k) Loss_gl (l) Loss_tri



(m) GT (n) SDF (o) Original (p) Loss_md (q) Loss_gl (r) Loss_tri

Figure 13: Output reconstructed meshes of three instances from original model and the models with three constraints: Mahalanobis Distance loss ($Loss\_md$), Group Latent Loss ($Loss\_gl$), Triplet-like Loss ($Loss\_tri$)

4.3.3 Plane



(a) GT (b) SDF (c) Original (d) Loss_md

(e) Loss_gl (f) Loss_gl_nr (g) Loss_tri (h) Loss_tri_nr

(i) GT (j) SDF (k) Original (l) Loss_md

(m) Loss_gl (n) Loss_gl_nr (o) Loss_tri (p) Loss_tri_nr

(q) GT (r) SDF (s) Original (t) Loss_md

(u) Loss_gl (v) Loss_gl_nr (w) Loss_tri (x) Loss_tri_nr

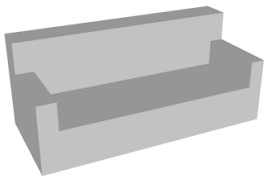
(y) GT (z) SDF () Original () Loss_md

() Loss_gl () Loss_gl_nr () Loss_tri () Loss_tri_nr

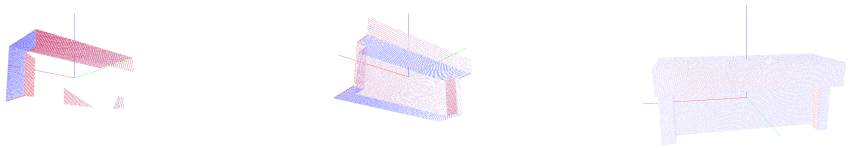
Figure 14: Output reconstructed meshes of three instances from original model and the models with three constraints: Mahalanobis Distance loss ($Loss\_md$), Group Latent Loss ($Loss\_gl$), Triplet-like Loss ($Loss\_tri$) and 2 additional experiment $Loss\_tri\_nr$ and $Loss\_gl\_nr$

4.4 Mesh with Constraint Loss

We add Mahalanobis Distance Loss during the optimization on latent codes of 3 partial observation from same sofa. The model is trained with Triplet-like Loss.

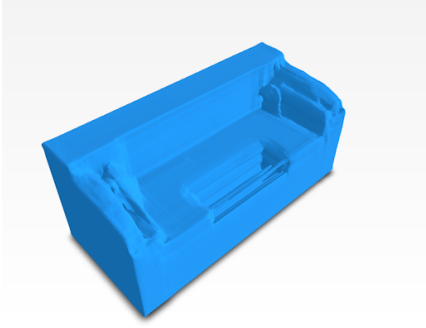


(a) Ground-truth Mesh

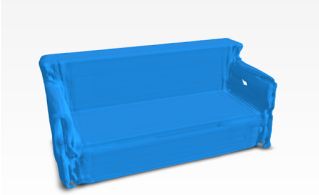


(b) Partial SDF Sample 1 (c) Partial SDF Sample 2 (d) Partial SDF Sample 3

Figure 15: The mesh of sofa and its 3 different partial SDF Samples

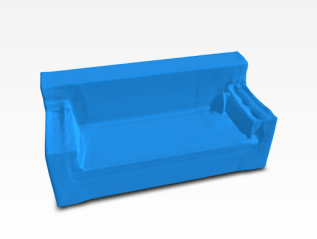


(a) Simply Stacked



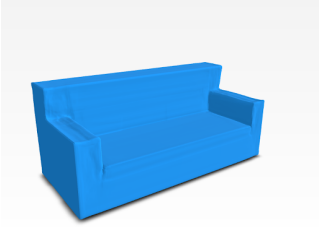
(b) Reconstruction partial 1

(c) Reconstruction partial 2

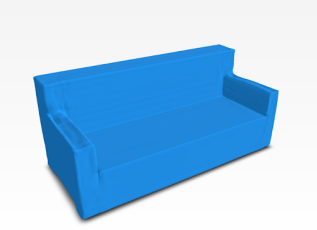


(d) Reconstruction partial 3

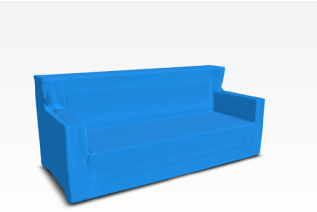
Figure 16: The reconstructions from the simply stacked SDF points from 3 partial samples and from separated 3 partial samples



(a) Reconstruction SDF Sample 1



(b) Reconstruction SDF Sample 2



(c) Reconstruction SDF Sample 3

Figure 17: The reconstruction with Mahalanobis Distance Loss

4.5 Reconstruction error

4.5.1 Sofa

Averaged Mesh Error	original	loss_md	loss_gl	loss_tri
Chamfer Distance	0.002365	**0.001842**	0.002063	0.002034

Table 11: Chamfer distance for the reconstructed meshes, evaluated from original model and the models with three constraints: Mahalanobis Distance loss (Loss_md), Group Latent Loss (Loss_gl), Triplet-like Loss (Loss_tri),Lower is better

4.5.2 lamp

Averaged Mesh Error	original	loss_md	loss_gl	loss_tri
Chamfer Distance	0.008010	**0.007235**	0.007574	0.007318

Table 12: Chamfer distance for the reconstructed meshes, evaluated from original model and the models with three constraints: Mahalanobis Distance loss (Loss_md), Group Latent Loss (Loss_gl), Triplet-like Loss (Loss_tri),Lower is better.

4.5.3 plane

Averaged Mesh Error	original	loss_md	loss_gl	loss_gl_nr*	loss_tri	loss_tri_nr*
Chamfer Distance	0.000987	0.000823	0.001161	0.001090	0.000777	**0.000731**

Table 13: Chamfer distance for the reconstructed meshes, evaluated from original model and the models with three constraints: Mahalanobis Distance loss (Loss_md), Group Latent Loss (Loss_gl), Triplet-like Loss (Loss_tri) and 2 additional experiment $Loss\_tri\_nr$ and $Loss\_gl\_nr$,Lower is better.

5 Conclusion

In our work, we proposed adapted models based on DeepSDF[1]. By using partial data as training input and at the same time adding three different constraint losses: Mahalanobis Distance Loss, Group Latent Loss, Triplet-like Loss, we can enforce the model to generate close latent vectors for different partials of one instance. We have shown that our models have can be more accurate for partial-to partial matching in latent space. With smaller reconstruction errors, the completion from single-view inputs has also been improved.

For future works, we can extend our work by leveraging latent space to get pose information for re-localization, so that the model could be applied for partial observation under non-canonical frame.

By using distributed architecture, the optimization of latent codes and the decoder could be more synchronized and more efficient. The idea of training on partial data with constraint loss could also be generalized for other models which work on latent space.

References

[1] J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. pages 165–174, 06 2019. doi:10.1109/CVPR.2019.00025.

[2] D. G. Lowe. Local feature view clustering for 3d object recognition. 1:I–I, 2001.

[3] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. pages 548–562, 2012.

[4] C. Steger. Similarity measures for occlusion, clutter, and illumination invariant object recognition. pages 148–154, 2001.

[5] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. pages 536–551, 2014.

[6] N. N. F. T. M. N. Johanna Wald, Armen Avetisyan. Rio: 3d object instance re-localization in changing indoor environments. *Proceedings IEEE International Conference on Computer Vision (ICCV)*, 2019.

[7] V. Sitzmann, M. Zollhoefer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. 32, 2019. URL https://proceedings.neurips.cc/paper/2019/file/b5dc4e5d9b495d0196f61d45b26ef33e-Paper.pdf.

[8] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. pages 652–660, 2017.

[9] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[10] A. Sinha, J. Bai, and K. Ramani. Deep learning 3d shape surfaces using geometry images. pages 223–240, 2016.

[11] O. Litany, A. Bronstein, M. Bronstein, and A. Makadia. Deformable shape completion with graph convolutional autoencoders. pages 1886–1895, 2018.

[12] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. pages 52–67, 2018.

[13] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. pages 1912–1920, 2015.

[14] G. Riegler, A. Osman Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. pages 3577–3586, 2017.

[15] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. 2018.

[16] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004. URL https://proceedings.neurips.cc/paper/2003/file/d3b1fb02964aa64e257f9f26a31f72cf-Paper.pdf.

[17] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. (arXiv:1512.03012 [cs.GR]), 2015.