Self-supervised Shape Completion via Involution and Implicit Correspondences Supplementary Document

Mengya Liu[1], Ajad Chhatkuli[1,3], Janis Postels[1], Luc Van Gool[1,3], and Federico Tombari[2]

[1] Computer Vision Lab, ETH Zurich
[2] Google, TU Munich
[3] INSAIT, Sofia University
{mengya.liu, ajad.chhatkuli, jpostels, vangool}@vision.ee.ethz.ch, tombari@google.com

I Overview

In this supplementary document, we provide additional details on the experiments, and network architectures as well as additional visualizations and dataset evaluations. We first provide additional experimental details in Section II including the training details and the hyper-parameters setup. Next, we detail the mesh completion process at inference time. We describe concretely the pre-processing steps for creating our PartialUDF dataset in Section III. Next, we provide more experimental results, and the correspondences analysis in Section II. Finally, we analyze the limitations of our method in Section V.

II Experimental Details

II.1 Architecture details and hyper-parameters

Our network architecture has two modules, the completion module and the template-based INR module ($\mathcal{T}$). The template-based INR module is composed of the warp function D and the INR decoder T, whereas the completion module includes the completion function $\mathcal{G}$ and the upsampler $\mathcal{U}$.

Template-based INR module. The warp function D has 10 layers of rnvp+lstm architecture [8,16], with the hidden size of 128, and the decoder T is parameterized with 4 MLP layers with the hidden size of 256, the last layer is a combination of *abs* and tanh functions to output the UDF values within $[0, 1]$. Moreover, similar to DeepSDF [10] that focuses more on the near-surface area, we also clamp the predicted UDF values d to $[0, 0.1]$.

Completion module. The completion function $\mathcal{G}$ [13] has two sub-networks, the encoder $\mathcal{E}$ and the decoder G. The encoder $\mathcal{E}$ is used to extract a 256-dim shape code c', with a pointnet++ [11] based set abstractions to extract a global feature, together with a transformer [15] to encode the local context. The decoder G takes as input the extracted feature c' and outputs 128 points Y, describing the missing part. Specifically, it first extracts the sparse point features, followed

by 3 MLP layers acting on the concatenation of the per-point feature and the shape code to generate the missing points Y. Finally, the generated Y points are concatenated with the input points X' and down-sampled to X_c consisting of 512 points by farthest point sampling (FPS). Next, the sub-module upsampler [13] consists of three steps of Snowflake Point Deconvolution (SPD). Each step involves splitting each point into multiple child points by duplication and adding variations. The upsampling factor is $[1, 2, 2]$, which outputs a denser point cloud with 2048 points.

Hyper-parameters.

Although, the SDF/UDF allows for shape representation superior to standard discrete representations, it does come with its own limitations. Specifically, considering the partial shape UDF field is not equal to the full shape UDF field, the far-away UDF samples may likely have the incorrect UDF values on the partial shape compared to those on the full shape. In order to mitigate its effects, we set the reconstruction loss weight to be larger on points closer to the surface, and smaller on points far away.

$$w_{\theta_\mathcal{T}} = \begin{cases} = 5e^{-d_i}, d_i <= 0.005 \\ = e^{-d_i}, d_i > 0.005 \end{cases} \tag{1}$$

where d_i denotes the initial UDF values. Thus, the reconstruction loss $\mathcal{L}_\phi$ is constructed,

$$\mathcal{L}_{\theta_\mathcal{T}} = \sum_{n \in \{2,4,6,8,10\}} w_\phi \mathcal{L}^n_{curr}(d^n_i, d_i) \tag{2}$$

$\mathcal{L}^n_{curr}$ denotes the progressive reconstruction loss between the predicted d^n_i from the n^{th} warp layer and d_i. We apply the loss in $2^{nd}, 4^{th}, 6^{th}, 8^{th}$, and 10^{th} layers.

Model complexity. We train the network for 2500 epochs with an initial learning rate of 0.0005 and the decay rate is 0.5 every 500 epochs. For each input, 2048 on-surface points and 4000 UDF samples are randomly sampled from the dataset. The batch size is set to 24, and the model is trained with 4 NVIDIA Titan RTX GPUs. The model size is nearly 7M, and the training of the model in the cars category costs nearly 1 day with a training set of around 3000 partial shapes.

II.2 Inference details

At the inference time, we predict the partial shape feature c' from the partial input (point cloud only) and later extract the full shape global feature c_X. It conditions the template-based INR module to generate a complete implicit surface. A detailed workflow outlining this process is depicted in Figure A. The partial input is passed through the completion function $\mathcal{G}$ and upsampler $\mathcal{U}$. These components collectively generate the complete shape representation, denoted as X, and extract the global shape code c_X. The UDF samples in the space here are voxel points in the unit box with a resolution of 256 [7] and predicted with the supervision of c_X. To obtain a dense surface mesh of the shape, we apply MeshUDF [7] which leverages the predicted UDF values.

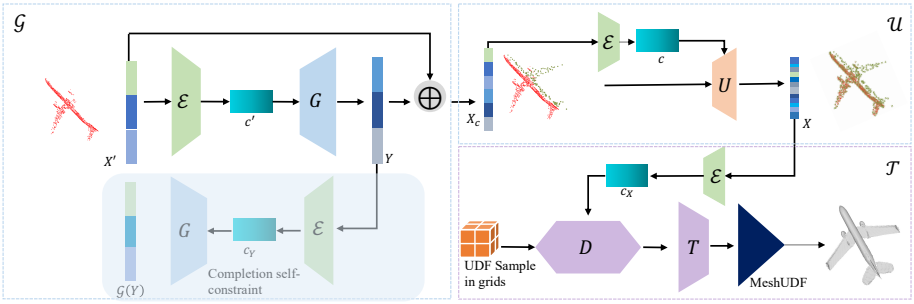

Fig. A: Inference time visualization. The figure shows the forward flow at the inference time. The partial shape is used to extract the partial shape code c' and the full shape global code c_X after completing the partial shape. We randomly sample points where we output the INR UDFs conditioned on c_X and generate the dense mesh through MeshUDF [7]. The blue box contains the part for completion self-constraint loss, which is not used during inference time.

III PartialUDF Dataset Pre-processing

We provide detailed dataset pre-processing steps here. Following DeepSDF [10], we prepare the UDF samples from synthetic objects in ShapeNetCore.v2 [2], which provide the complete 3D meshes. Similar to DeepSDF, we employ virtual cameras positioned around the objects in a unit sphere. However, instead of using 100 virtual cameras, we opt for 30 equally spaced camera positions. Moreover, DeepSDF densely samples surface points from all 100 views, and computes the Signed Distance Field (SDF) around the points. We instead sample surface points and compute the Unsigned Distance Field (UDF) for each view, thus generating the partial shapes and the UDF field for the partial shapes. 30,000 surface points from each view are selected, and another set of 50,000 points are randomly sampled in the surrounding space to compute the UDF field. The surface points are sampled more aggressively near the surface. To draw a conclusion, for each shape instance, there are 30 partial shapes, each consisting of dense surfaces with 30,000 surface points and 50,000 UDF samples. We additionally provide the normalization parameters for each shape. Note that, we randomly select only 6 partial views for each instance during training and 2 partial shapes for the test throughout each experiment, while ensuring that these shapes are in different batches during the training. Furthermore, we compare our dataset with the popular 3D-EPN dataset [6], which prepares the partial shapes by generating the depth maps from random views. Both datasets come from the CAD models in ShapeNetCore.v2 [2]. We measure the missing rates of some categories in both datasets, results are presented in Table A. We see our dataset has similar missing rates compared with 3D-EPN dataset. Except for the planes category, our dataset has smaller missing rates as 3D-EPN [6].

In Figure B, we present visualizations of various partial observations from the object. The first column shows the camera position (boxed) and the central

Categories	cars	planes	sofas	chairs	tables
3D-EPN [6]	65.89	56.96	64.65	63.21	63.54
PartialUDF-Shapenet	64.26	60.55	63.46	60.96	57.14

Table A: Missing rates (%) of various categories in 3D-EPN [6] and our PartialUDF-Shapenet datasets.

localized object. The extracted partial shapes are presented in the right columns from 6 random views for each object. The figure implies the varying degrees of partiality observed across different views.

Moreover, in order to explore the applicability of our method to non-rigid shapes, we also use the DFaust dataset [1] which provides both dense mesh objects and ground-truth correspondences. Following a similar pre-processing procedure as applied to the PartialUDF-Shapenet dataset, we obtained partial inputs for a total of 9 sequences in the DFaust dataset, among which 7 sequences are utilized for training and the rest 2 sequences for test. These sequences contain slow but real, dynamic human body motions. Figure D provides a visual representation of the prepared partial shapes from various viewpoints. The partial shapes from the views are always quite sparse (the missing rate is around 72%), which makes it much more interesting dataset for the completion task.

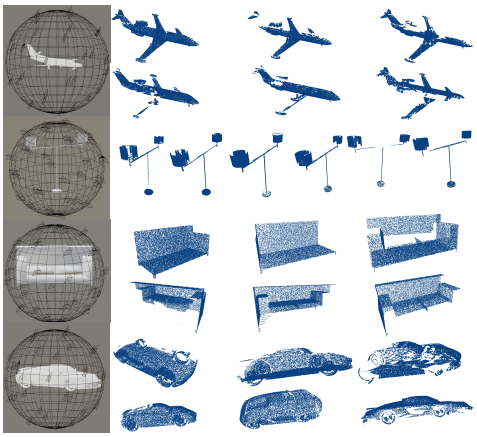

Fig. B: PartialUDF-Shapenet Dataset visualization. The first column shows the camera setup (boxed) and the object mesh, followed by the visualization of partial shapes from different views.

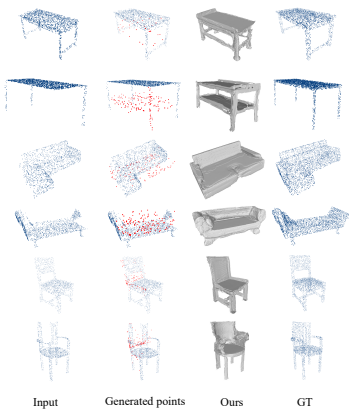

Fig. C: Visualization results of failure cases. We show some failure cases such as the generation of outlier points (in red) and/or bad completion (in mesh).

Fig. D: Visualizations of partial observations of DFaust dataset [1]. The figure shows the partial shapes of the DFaust dataset from various sequences and from different views.

IV More Experimental Results

We show some additional experimental results including visualizations in this section.

IV.1 Correspondences Analysis

We further explore the capability of our method in improving the dense correspondences. From Table 2 in the main paper, we decrease the dense correspondences $\ell 2$ errors from 0.34 (INR_only) to 0.28 in the DFAUST dataset [1], showing that by gradually completing the template space, we also improve the dense correspondences. We provide the qualitative results in Figure E. Note the diverse body poses we are able to complete. We show the template shapes provided by our method as well as the template-based INR model (INR_only) in the green circle with colors representing the correspondences. We also display the completed shapes from two methods with colors, the same colors denote the corresponding points and keep the same as in the template shape. We observe that our method generates a mean shape of the dataset by gradually completing the template space, which is beneficial for dense correspondences. In contrast, we can obviously see the color inconsistency in the INR_only method.

IV.2 More experimental results on ScanNet Dataset

we furthermore test our method on another real-world dataset, ScanNet [5] bathtub, bed, and lamp categories provided by PatchCompletion [12]. During inference time, we extract the partial point clouds from their voxelized representation as the input. Our method is trained on the MVPS-Shapenet categories. We compare our method with the supervised autoencoder 3D-EPN [6], autoregressive model AutoSDF [9] and supervised patch-based method PatchComplete [12]. We measure the CD errors following PatchComplete, results are presented in Table B. Results from the other methods are obtained from PatchComplete [12]. Our method defeats the supervised methods in both bathtub and lamp categories. More visualization results are in Figure F. Our method completes accurate shapes while PatchComplete often fails to complete accurately.

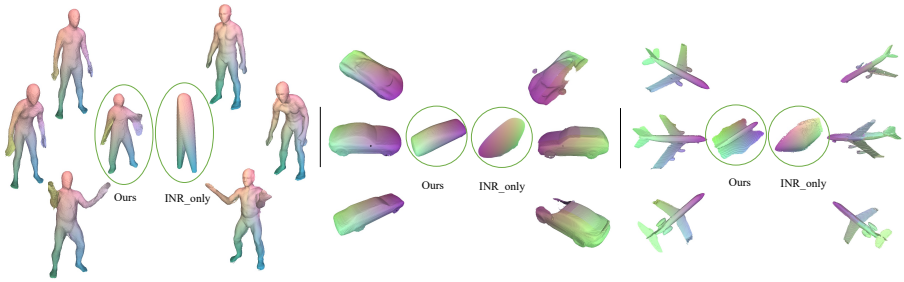

Fig. E: Qualitative results of dense correspondences compared with INR_only method. We provide the generated template shapes in the green circles, as well as the completed meshes for both methods. The same color represents the corresponding points across various shapes similar to [16].

Methods type	3D-EPN [6] sp.	AutoSDF [9] sp.	PatchComplete [12] sp.	Ours ssp.
Bathtub	7.56	7.84	6.77	**6.55**
Bed	7.76	7.91	**7.24**	7.50
Lamp	14.27	11.17	9.42	**9.34**

Table B: Quantitative results on Scannet dataset. We present CD errors compared with 3D-EPN [6], AutoSDF [9], and PatchComplete [12]. Our method achieves compatible results.

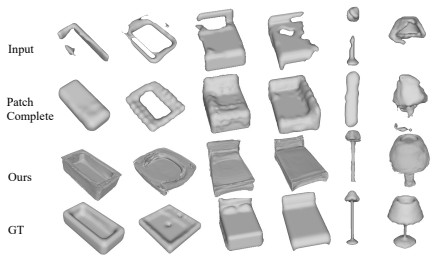

Fig. F: Qualitative results on Scannet dataset. Compared with PatchComplete [12], our method generates accurate shapes.

Completeness	F1	CD
28%	84.57	1.76
65%	87.84	0.44

Fig. G: Evaluation of the method performance regarding different completeness in PartialUDF DFaust dataset.

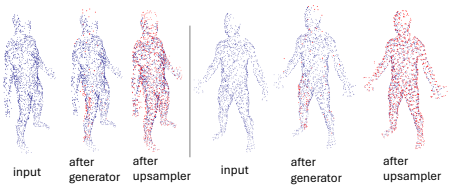

Fig. H: Completion module performance on almost complete inputs. The blue points represent the input point cloud, red points are generated points after the generator (completion function) and the upsampler respectively.

IV.3 More quantitative results

We further provide more quantitative results on PartialUDF-Shapenet and PartialUDF -DFaust dataset in Figure I,J, K. Compared with other methods, our method generates clean and full meshes from the partial point clouds, while P2C [4] generates noisy point clouds, and cannot fill in the missing areas well. We also show detailed points generation from the completion module, as can be seen, the point cloud is completed under the supervision of the template space.

IV.4 Ablation study with almost complete shape

We also explore the ability of the involution function when dealing with almost complete shapes. We evaluate it with the setup in PartialUDF-DFAUST with higher completeness of 65% (28% in the main paper experiments)). Note that, the completeness of 65% is the average completeness of the trainset, which includes the partial shapes with completeness from around 30% to 90%. Results are shown in Table G. When the completeness improved, both F1 score and CD errors present better performance. We additionally show the visualization of the generated points with almost complete shapes in Figure H.

We recall that $\mathcal{G}(X')$ provides the missing part complementary to X'. Thus, when X' is almost complete, $\mathcal{G} \circ \mathcal{G}(X')$ should struggle to predict correctly, under such definition of $\mathcal{G}$. However, in practice, we do not enforce $\mathcal{G}(X')$ to *not* include points in X'. Thus, for such a set of points X' which is almost complete, $\mathcal{G}$ can learn to predict missing points as well as part of the initial points (part trivial solution). Moreover, the reconstruction loss on INR T ensures that the existing input shape is preserved in $X' \cup \mathcal{G}(X')$.

V Limitations

While our method demonstrates the capability to complete shapes in several categories in a self-supervised manner, it is important to acknowledge that self-supervised shape completion remains a highly challenging task. For example, we

present some failure cases in Figure C, where we see that incorrect shape parts are hallucinated by our method, resulting in the loss of performance. Moreover, even when the generated points are of high quality, there are instances where the template INR model becomes the limiting factor. As a consequence, the self-supervised shape completion and correspondences in a wider range of categories may require additional priors. Another limiting factor is the use of UDF for shape representation which is known to be inferior to SDF, suffering from noise and shape artifacts [7]. However, we found it rather challenging in order to obtain consistent SDF values starting from the UDF values. Note that it is not possible to have SDF values of partial shapes as the inside-outside definition may not be available.

Fig. I: Qualitative results on PartialUDF-DFaust dataset. We present the completed shapes generated from various methods with different partial point clouds.

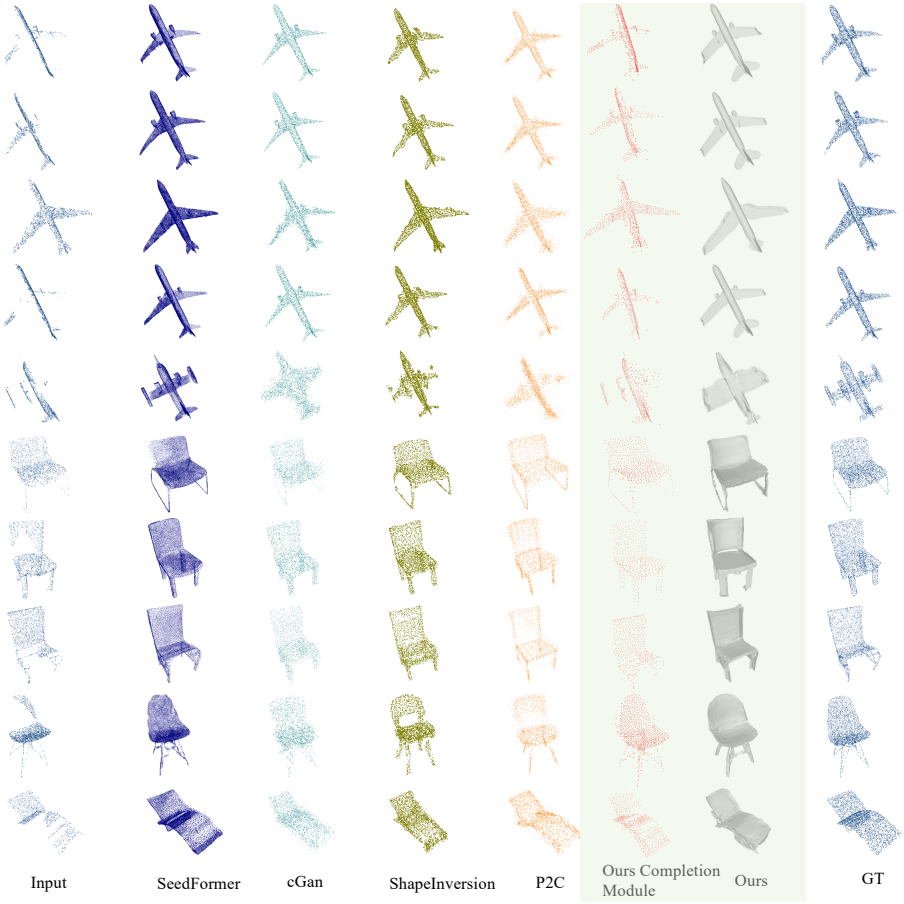

Fig. J: More visualization results on PartialUDF-Shapenet dataset planes, chairs categories. Compare with supervised SeedFormer [17], unpaird cGan [3], pretrained-treeGan based ShapeInversion [14], self-supervised P2C [4] methods.

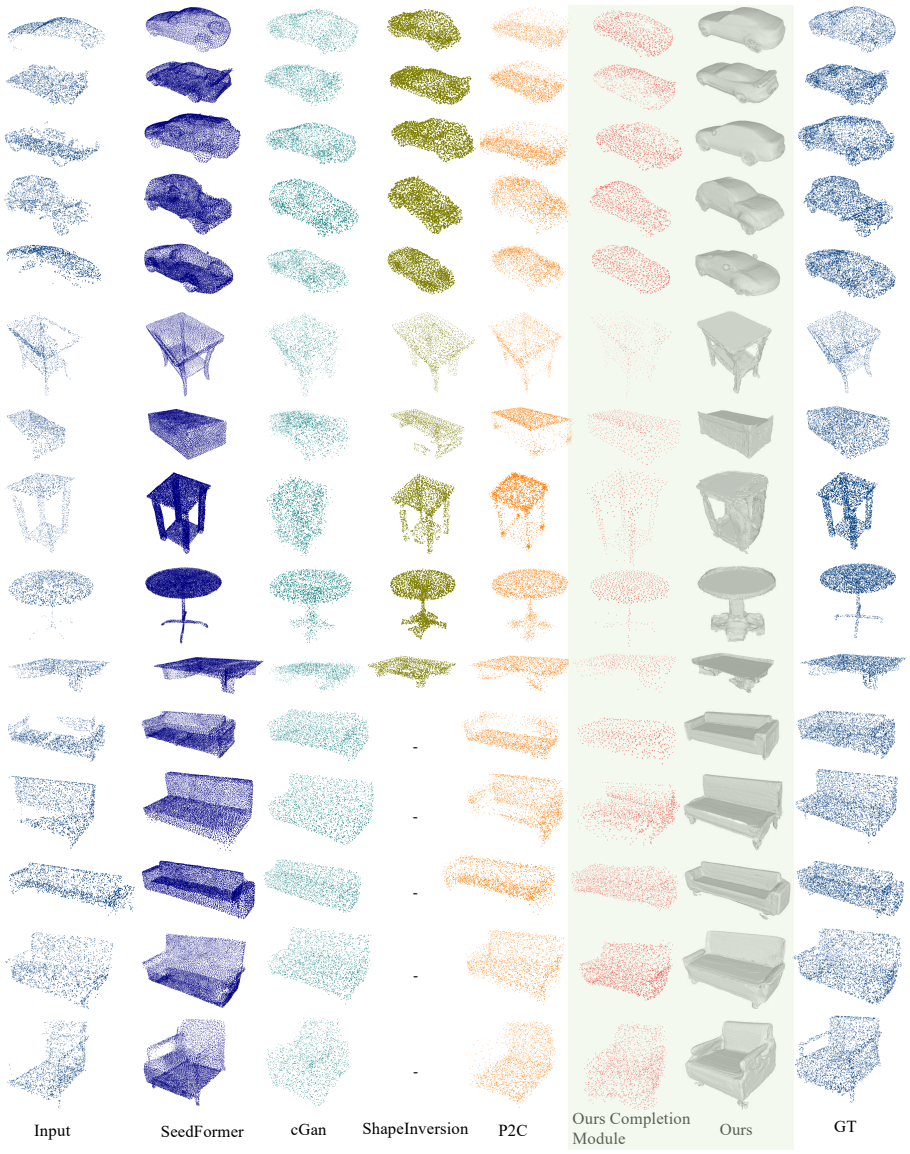

Fig. K: More visualization results on PartialUDF-Shapenet dataset cars,tables and sofas categories. Compare with supervised SeedFormer [17], unpaird cGan [3], pretrained-treeGan based ShapeInversion [14], self-supervised P2C [4] methods.

References

1. Bogo, F., Romero, J., Pons-Moll, G., Black, M.J.: Dynamic faust: Registering human bodies in motion. In: CVPR (2017)
2. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
3. Chen, X., Chen, B., Mitra, N.J.: Unpaired point cloud completion on real scans using adversarial training. arXiv preprint arXiv:1904.00069 (2019)
4. Cui, R., Qiu, S., Anwar, S., Liu, J., Xing, C., Zhang, J., Barnes, N.: P2c: Self-supervised point cloud completion from single partial clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14351–14360 (2023)
5. 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 conference on computer vision and pattern recognition. pp. 5828–5839 (2017)
6. Dai, A., Ruizhongtai Qi, C., Nießner, M.: Shape completion using 3d-encoder-predictor cnns and shape synthesis. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5868–5877 (2017)
7. Guillard, B., Stella, F., Fua, P.: Meshudf: Fast and differentiable meshing of unsigned distance field networks. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III. pp. 576–592. Springer (2022)
8. Liu, M., Chhatkuli, A., Postels, J., Gool, L.V., Tombari, F.: Unsupervised Template Warp Consistency for Implicit Surface Correspondences. Computer Graphics Forum (2023). https://doi.org/10.1111/cgf.14745
9. Mittal, P., Cheng, Y.C., Singh, M., Tulsiani, S.: Autosdf: Shape priors for 3d completion, reconstruction and generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 306–315 (2022)
10. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 165–174 (2019)
11. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413 (2017)
12. Rao, Y., Nie, Y., Dai, A.: Patchcomplete: Learning multi-resolution patch priors for 3d shape completion on unseen categories. Advances in Neural Information Processing Systems **35**, 34436–34450 (2022)
13. Xiang, P., Wen, X., Liu, Y.S., Cao, Y.P., Wan, P., Zheng, W., Han, Z.: Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 5499–5509 (2021)
14. Zhang, J., Chen, X., Cai, Z., Pan, L., Zhao, H., Yi, S., Yeo, C.K., Dai, B., Loy, C.C.: Unsupervised 3d shape completion through gan inversion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1768–1777 (2021)
15. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 16259–16268 (2021)

16. Zheng, Z., Yu, T., Dai, Q., Liu, Y.: Deep implicit templates for 3d shape representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1429–1439 (2021)
17. Zhou, H., Cao, Y., Chu, W., Zhu, J., Lu, T., Tai, Y., Wang, C.: Seedformer: Patch seeds based point cloud completion with upsample transformer. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III. pp. 416–432. Springer (2022)