Supplementary Material of Single Depth-image 3D Reflection Symmetry and Shape Prediction

Zhaoxuan Zhang, Bo Dong*, Tong Li, Felix Heide, Pieter Peers, Baocai Yin*, Xin Yang*

This supplemental document provides additional implementation details, ablation studies, and qualitative comparisons to state-of-the-art methods to support the submission of "Single Depth-image 3D Reflection Symmetry and Shape Prediction".

1. Formal definitions of quantitative metrics (CD and MD)

The Chamfer Distance (CD) [7] and mesh-to-mesh symmetric distance (MD) [1] are used to compare competing methods quantitatively. For completeness, we provide the mathematical definitions of these two metrics.

Given two 3D point sets X and Y, the CD is defined as:

$$CD(X,Y) = \frac{1}{2N^X}\sum_{x\in X}\min_{y\in Y}\|x-y\|_2 + \frac{1}{2N^Y}\sum_{y\in Y}\min_{x\in X}\|x-y\|_2, \quad (1)$$

where N^X and N^Y indicate the numbers of points in the point sets X and Y, respectively.

Given two meshes S_1 and S_2, the MD is defined as:

$$MD(S_1,S_2) = \frac{1}{2N^{S_1}}\sum_{p_1\in V^{S_1}} e(p_1,S_2) + \frac{1}{2N^{S_2}}\sum_{p_2\in V^{S_2}} e(p_2,S_1), \quad (2)$$

$$e(p,S) = \min_{p'\in S}\|p-p'\|_2, \quad (3)$$

where N^{S_1} and N^{S_2} are the number of vertices in the mesh S_1 and S_2, respectively; V^{S_1} and V^{S_2} are the set of vertices from S_1 and S_2, respectively.

2. Implementation Details of the Viewpoint Selection Module

Reinforcement Learning (RL) Agent The architecture of the RL agent consists of an encoder, decoder, critic, and actor heads. We adopt a commonly used 3D point encoder [3, 4] which extracts features from the input point cloud for downstream processing by the actor and critic. In particular, the actor head predicts a viewpoint defined by [θ, ϕ] based on an input point cloud P with $8,192$ 3D points. Mathematically, the RL agent is defined as follows:

$$\begin{aligned}\hat{R} &= \zeta^{br}_{256,1}(\text{FH}(F_2^V)),\\ [\theta,\phi] &= \text{Tanh}(\zeta^{br}_{256,2}(\text{SH}(F_2^V))),\\ F_2^V &= \zeta^{br}_{1024,512}(\zeta^{br}_{1024,1024}(F_1^V)),\\ F_1^V &= \text{MP}(\zeta^{b}_{128,1024}(\zeta^{br}_{64,128}(\zeta^{br}_{3,64}(P)))),\end{aligned} \quad (4)$$

where $\hat{R}$ is the output of the critic head, estimating total rewards in the future based on current point cloud P; The functions FH$(\cdot)$ and SH$(\cdot)$ represent the first and second half of an input vector; $\zeta^{br}_{m,n}$ is a linear layer with input channel number m, output channel number n followed by a Batch Normalization (BN;indicated with a b superscript) and ReLU activation function (indicated with an r superscript). MP$(\cdot)$ is a max-pooling function.

RL Agent Training The RL agent is trained by using the actor-critic Proximal Policy Optimisation (PPO) algorithm [6] with the following settings: T-horizon is set to 30, the clipping parameter is 0.2, the discount factor is 0.9 and 4 epochs in each PPO update. During the training, the weights of other components of ISCNet are fixed, and we only update the weights of the RL agent.

3. Implementation Details of the 2D Inpainting Module

The 2D inpainting module comprises three components: mask generation, normal map inpainting, and depth-map inpainting. For completeness, we include the formal definitions of each module:

Mask Generation The mask generation module's main component extracts silhouette information via a variant of

*Co-corresponding authors.

the U-Net architecture [5]. Given an initial input depth map, D^I, and a symmetric depth map pair D_i and D'_i, the silhouette extraction component outputs a pair of silhouette maps K_i and K'_i. Formally, the process is defined as follows:

$$\begin{aligned} K_i &= G^M([F_5^M, F_1^M, F_1^{M'}]), \\ K'_i &= G^{M'}([F_5^M, F_1^M, F_1^{M'}]), \\ F_5^M &= M_2^D([F_4^M, F_2^M]), \\ F_4^M &= M_1^D(F_3^M), \\ F_3^M &= M_2^E(F_2^M), \\ F_2^M &= M_1^E([F_1^M, F_1^{M'}]), \\ F_1^M &= E^M([D_i, D^I]), \\ F_1^{M'} &= E^{M'}([D'_i, D^I]), \end{aligned} \tag{5}$$

where $[\cdot,\cdot,\cdot]$ indicates a channel-wise concatenation operation. Intuitively, it has an encoder with two input heads, E^M and $E^{M'}$, and two encoding blocks, M_1^E and M_2^E. On the decoder side, it has a similar pattern, two decoding blocks, M_1^D and M_2^D, and two output heads G^M and $G^{M'}$. Mathematically, the components are defined as:

$$G^M(\cdot) := \sigma(\psi_{31}(\mathscr{U}(\psi_{31}^{br}(\mathscr{U}(\psi_{31}^{br}(\mathscr{U}(\cdot))))))), \tag{6}$$
$$M_x^D(\cdot) := \psi_{31}^{br}(\mathscr{U}(\psi_{31}^{br}(\mathscr{U}(\cdot)))), \tag{7}$$
$$M_x^E(\cdot) := \psi_{32}^{br}(\psi_{32}^{br}(\cdot)), \tag{8}$$
$$E^M(\cdot) := \psi_{52}^{br}(\psi_{52}^{br}(\psi_{72}^{r}(\cdot))), \tag{9}$$

where ψ_{ks}^{br} is a $k \times k$ convolution layer with stride s followed by a Batch Normalization (BN; indicated with a b superscript) and ReLU activation function (indicated with an r superscript); $\sigma(\cdot)$ is a SoftMax function; $\mathscr{U}(\cdot)$ indicates the up-sampling operation with stride 2. Note that G^M and $G^{M'}$ are identical in architecture but with different weights. Idem for E^M and $E^{M'}$.

Normal Inpainting. As discussed in the main paper, the normal inpainting submodule $\mathscr{N}$ takes as input a pair of partial normal maps N_i and N'_i corresponding to symmetric viewpoints v_i and v'_i as well as their corresponding masks M_i and M'_i, and outputs a pair of completed normal maps $\hat{N}_i$ and $\hat{N}'_i$ respectively:

$$\hat{N}_i, \hat{N}'_i = \mathscr{N}([N_i, \mathbf{1} - M_i], [N'_i, \mathbf{1} - M'_i]). \tag{10}$$

The network $\mathscr{N}$ has similar architecture as the one used for the mask generator and which is formally defined as:

$$\begin{aligned} \hat{N}_i &= G^N([F_5^N, F_1^N, F_1^{N'}]), \\ \hat{N}'_i &= G^{N'}([F_5^N, F_1^N, F_1^{N'}]), \\ F_5^N &= N_2^D([F_4^N, F_2^N]), \\ F_4^N &= N_1^D(F_3^N), \\ F_3^N &= N_2^E(F_2^N), \\ F_2^N &= N_1^E([F_1^N, F_1^{N'}]), \\ F_1^N &= E^N([N_i, \mathbf{1} - M_i]), \\ F_1^{N'} &= E^{N'}([N'_i, \mathbf{1} - M'_i]). \end{aligned} \tag{11}$$

Unlike the mask generation module, the normal inpainting submodule $\mathscr{N}$ leverages partial convolution layers [2] in both the encoder and decoder. Formally, the operators are defined as:

$$G^N(\cdot) := \phi_{31}(\mathscr{U}(\phi_{31}^{br}(\mathscr{U}(\phi_{31}^{br}(\mathscr{U}(\cdot)))))), \tag{12}$$
$$N_x^D(\cdot) := \phi_{31}^{br}(\mathscr{U}(\phi_{31}^{br}(\mathscr{U}(\cdot)))), \tag{13}$$
$$N_x^E(\cdot) := \phi_{32}^{br}(\phi_{32}^{br}(\cdot)), \tag{14}$$
$$E^N(\cdot) := \phi_{52}^{br}(\phi_{52}^{br}(\phi_{72}^{r}(\cdot))), \tag{15}$$

where ϕ_{ks}^{br} is a $k \times k$ a partial convolution layer [2] with stride s followed by a Batch Normalization (BN) and a ReLU activation function. Note that G^N and $G^{N'}$ are identical in architecture but with different weights. Idem for E^N and $E^{N'}$.

Depth Inpainting. The depth inpainting submodule, $\mathscr{D}$, takes as input a pair of partial depth maps D_i and D'_i corresponding to symmetric viewpoints v_i and v'_i as well as their corresponding masks (M_i and M'_i) and inpainted normal maps ($\hat{N}_i$ and $\hat{N}'_i$), and outputs a pair of completed depth maps $\hat{D}_i$ and $\hat{D}'_i$ and corresponding confidence maps C_i and C'_i:

$$\begin{aligned} (\hat{D}_i, C_i), (\hat{D}'_i, C'_i) = \mathscr{D}(&\left[D_i, \hat{N}_i, \mathbf{1} - M_i\right], \\ &\left[D'_i, \hat{N}'_i, \mathbf{1} - M'_i\right]). \end{aligned} \tag{16}$$

The architecture of $\mathscr{D}$ (similar to $\mathscr{N}$) uses a variant of U-Net. Formally:

$$\begin{aligned}
\hat{D}_i, C_i &= F_o[:,0,:,:], \sigma(F_o[:,1,:,:]),\\
\hat{D}'_i, C'_i &= F'_o[:,0,:,:], \sigma(F'_o[:,1,:,:]),\\
F_o &= G^D([F_5^D, F_1^D, F_1^{D'}]),\\
F'_o &= G^{D'}([F_5^D, F_1^D, F_1^{D'}]),\\
F_5^D &= D_2^D([F_4^D, F_2^D]),\\
F_4^D &= D_1^D(F_3^D),\\
F_3^D &= D_2^E(F_2^D),\\
F_2^D &= D_1^E([F_1^D, F_1^{D'}]),\\
F_1^D &= E^D(D_i, \hat{N}_i, \mathbf{1} - M_i]),\\
F_1^{D'} &= E^{D'}([D'_i, \hat{N}'_i, \mathbf{1} - M'_i]). \qquad (17)
\end{aligned}$$

In terms of architecture, all the layers of the depth inpainting module are the same as for the normal inpainting module. Finally, F_o and F'_o have the same dimension, batch_size $\times$ $2 \times h \times w$. $F_o[:,x,:,:]$ indicates the x channel of F_o.

4. Additional Qualitative Evaluation

Figure 2 and Figure 3 provide additional qualitative comparisons for shape completions obtained with ISCNet versus competing methods. These results confirm that ISCNet offers the 'cleanest' shape completion for all eight tasks.

5. Additional Ablation Studies

We conduct two additional ablation studies to better understand the impact of the viewpoint selection and the number of iterations.

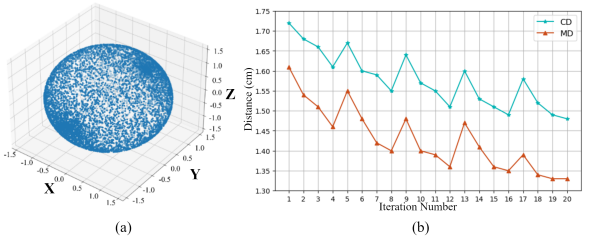
Impact of Viewpoint Selection. To demonstrate the effectiveness of the viewpoints selected by our RL agent, we conduct the following two experiments: shape reconstruction with four pairs of a). random reflection viewpoints; b). pre-defined reflection viewpoints. For the first experiment, we use four random viewpoints and their corresponding reflection viewpoints w.r.t. an estimated symmetry plane to replace the ones chosen by the RL agent. The experimental results are reported in Table 2, denoted by Random viewpoints I_x. The I_x means the iteration x, or shape reconstruction with the $(x\%4)$-th estimated symmetry plane. In the second experiment, we pre-define 4 viewpoints which are defined by 4 pairs of θ and ϕ, $[45,45], [135,45], [45,-45], [135,-45]$. We use the four viewpoints and their corresponding reflection viewpoints w.r.t. an estimated symmetry plane to replace the 4 pairs of reflection viewpoints selected by our RL agent. The results are reported in Table 2, Pre-defined viewpoints I_x.

Based on Table 2, our RL-based viewpoint selection method outperforms both random and pre-defined viewpoints schemes. In addition, the results of randomly selected viewpoints show that iterative refinement cannot improve the reconstructive quality progressively, indicating that viewpoint selection is essential for improving the symmetry plane estimation quality.

In Figure 4, we show one shape completion example with pre-defined and our RL-based viewpoint selection schemes, where the reconstruction results and 4 pairs of masks are shown under the corresponding 4 pairs of viewpoints. In each mask, the black color indicates the areas that need to be reconstructed. As we can see, our RL-based viewpoint selection scheme offers significantly better reconstruction quality than the one generated by pre-defined viewpoints; CD and MD are improved by 0.17 and 0.12, respectively. More experimental results with different viewpoint selection schemes are provided in Figure 7; our approach constantly offers the best shape completion results.

Number of Refinement Iterations. We empirically determine that 20 iterations strike a good balance between computational cost versus gain in accuracy. In our main paper, we report the error number of 1 to 20 iterations in Table 1 and Figure 6. For completeness, we include in Table 2, Figure 5 and Figure 6 the errors for 21 to 28 iterations.

Based on Figure 5a, we can see that the sixth and seventh symmetry plane estimate only offer marginally better performance than the fifth one. In terms of CD and MD, as shown in Figure 5b, the $20th$ iteration gives the best performance; this is also reflected in Table 2. Note that, compared to 20 iterations, the 8 extra iterations, *i.e.*, two extra symmetry plane estimates, require an additional 23.6 seconds of computations with little gain in accuracy. Figure 1b reports that CD and MD improve with each iteration, except briefly after each 4th-iteration reset. It is noteworthy that the overall reconstruction results of the 4th iteration already surpass those of all competing methods.



(a) (b)

Figure 1. (a) The distribution of selected viewpoints. (b) Impact of the number of iterations.

6. Time and Memory Statistics

We present a comprehensive analysis of inference time and GPU memory usage for our method compared to other benchmarked approaches, as detailed in Table 1. Notably, our method allocates a substantial portion of inference time toward the iterative generation of depth images and normal maps.

	Inf. time (s)	GPU Mem. (MB)
PCN	0.17	1,035
MSN	0.67	1,249
PoinTr	0.67	7,315
VRCNet	0.06	1,149
SnowF	0.24	1,235
VE-PCN	0.05	1,661
IFNet	22.4	10,075
Front2Back	0.53	2,277
Ours (20 iterations)	81.2	3,972
⊢*Networks*	21.3	-
⊢*Maps Generation*	42.4	-
⊢ *Point Cloud Merging & Labeling*	17.5	-

Table 1. Comparisons in inference time and GPU memory costs.

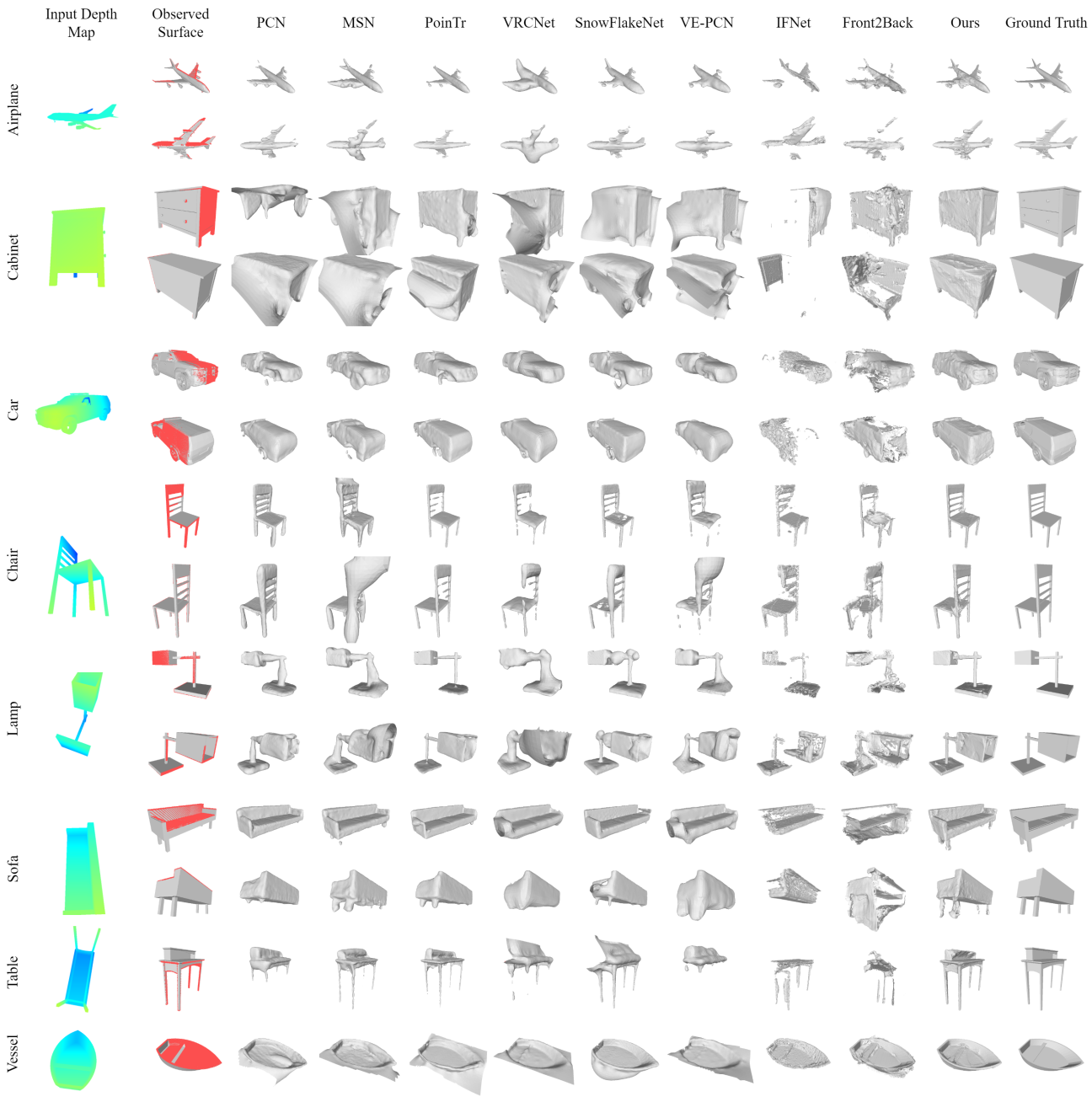


Figure 2. Additional visual comparisons of shape completions obtained with ISCNet versus prior work.

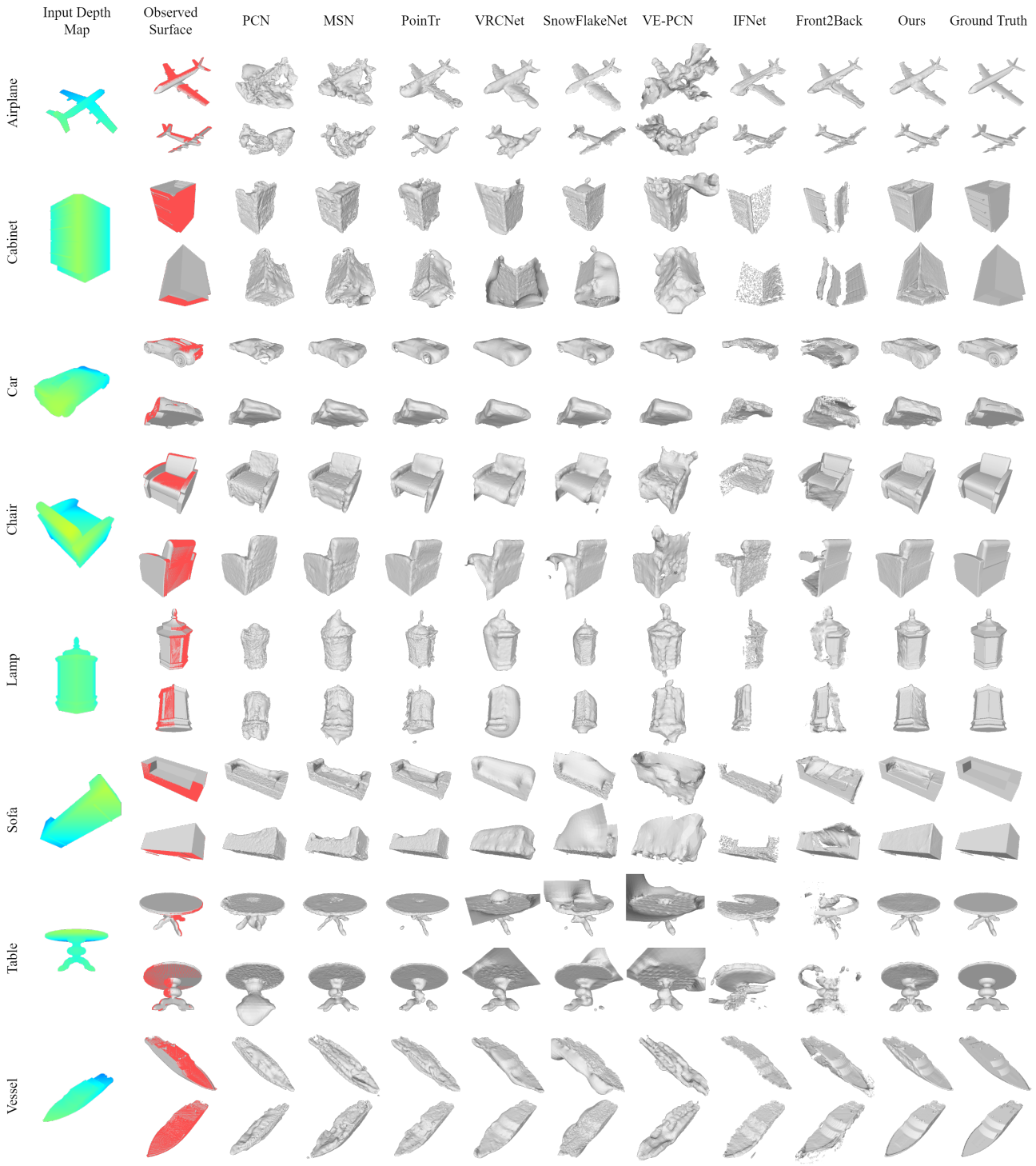


Figure 3. Additional visual comparisons of shape completions obtained with ISCNet versus prior work.

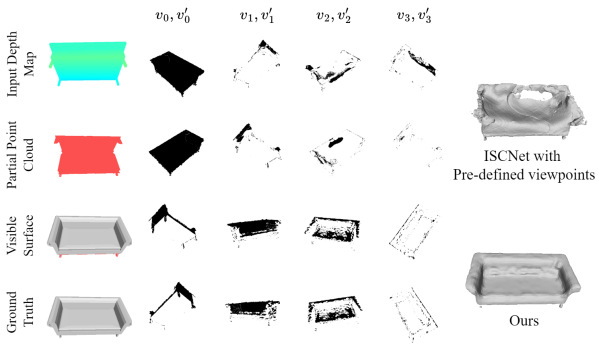
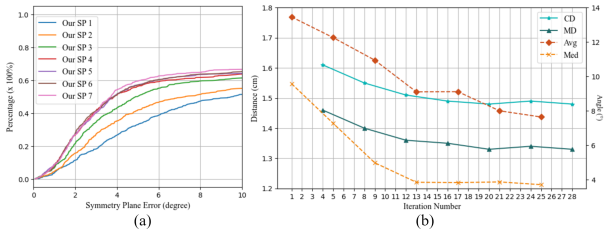
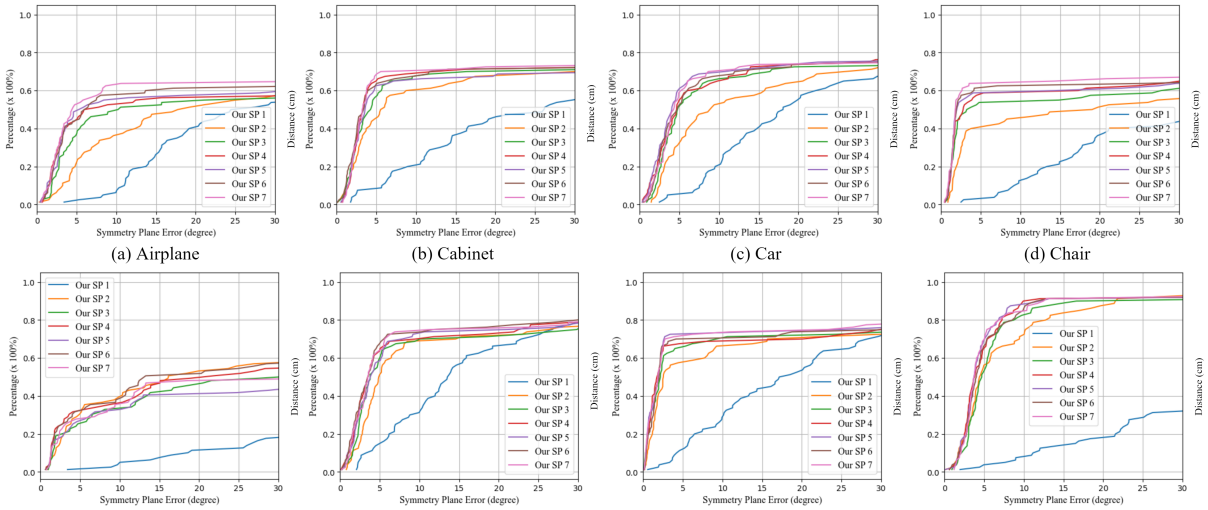


Figure 4. Visual comparisons of shape completions of ISCNet with pre-defined and our RL-based viewpoint selection schemes. The masks under different selected viewpoints are shown, where the black color indicates the areas that need to be completed.



(a) (b)

Figure 5. Impact of the number of iterations on (a) the accuracy of the symmetry plane estimation, and (b) the shape reconstruction MD/CD accuracy and average/median symmetry plane estimation errors.



(a) Airplane (b) Cabinet (c) Car (d) Chair

(e) Lamp (f) Sofa (g) Table (h) Vessel

Figure 6. Impact of the number of iterations on the accuracy of the symmetry plane estimation for each shape in ShapeNet.

$CD/MD(\times 10^2)$	*Plane*	*Cabinet*	*Car*	*Chair*	*Lamp*	*Sofa*	*Table*	*Vessel*	*Average*
Random viewpoints I_4	2.24/1.00	3.38/2.05	3.95/1.60	3.03/1.70	1.90/1.57	3.79/1.95	3.75/2.21	1.96/1.20	3.00/1.66
Random viewpoints I_8	3.06/1.05	3.18/1.85	4.36/1.60	2.97/1.63	2.08/1.60	3.71/1.86	3.54/2.17	2.82/1.19	3.21/1.62
Random viewpoints I_{12}	2.51/1.02	3.17/1.86	4.20/1.54	3.09/1.61	1.94/1.51	3.75/1.75	3.51/2.11	2.61/1.22	3.10/1.58
Random viewpoints I_{16}	2.87/1.06	3.37/1.74	4.26/1.56	3.04/1.56	1.83/1.58	3.59/1.77	3.53/2.11	2.71/1.21	3.15/1.57
Random viewpoints I_{20}	2.55/1.03	3.19/1.75	4.33/1.56	2.79/1.50	2.04/1.59	3.62/1.79	3.56/2.13	2.54/1.18	3.08/1.57
Pre-defined viewpoints I_4	0.86/0.86	2.36/1.80	1.78/1.42	1.88/1.47	1.67/1.37	1.94/1.73	1.95/1.94	1.19/1.13	1.70/1.46
Pre-defined viewpoints I_8	0.85/0.85	2.21/1.77	1.75/1.41	1.85/1.43	1.60/1.44	1.97/1.73	1.90/1.94	1.16/1.13	1.66/1.46
Pre-defined viewpoints I_{12}	0.88/0.88	2.18/1.76	1.71/1.35	1.79/1.36	1.63/1.43	1.90/1.69	1.88/1.91	1.12/1.12	1.63/1.43
Pre-defined viewpoints I_{16}	0.85/0.85	2.21/1.72	1.69/1.40	1.76/1.41	1.72/1.47	1.94/1.71	1.87/1.91	1.14/1.12	1.64/1.43
Pre-defined viewpoints I_{20}	0.86/0.87	2.15/1.65	1.68/1.41	1.79/1.43	1.88/1.53	1.86/1.67	1.85/1.89	1.12/1.12	1.65/1.45
Iteration 20 (Ours)	**0.82**/0.84	**2.04**/1.63	**1.56/1.35**	1.54/1.39	1.30/1.34	1.84/**1.53**	1.67/1.54	**1.04/1.00**	**1.48/1.33**
Iteration 24	0.83/**0.83**	2.13/1.67	1.58/1.36	**1.51/1.38**	**1.27/1.33**	1.89/1.58	**1.65/1.50**	1.04/1.00	1.49/1.33
Iteration 28	0.82/0.83	2.08/**1.61**	1.58/1.36	1.53/1.38	1.32/1.41	**1.83**/1.54	1.67/1.55	1.04/1.02	1.48/1.34

Table 2. Additional ablation studies: ISCNet with random viewpoints, pre-defined viewpoints, and refinement with extra iterations. The results show our RL-based viewpoint selection scheme offers the best performance, and extra refinement iterations do not lead to better results.

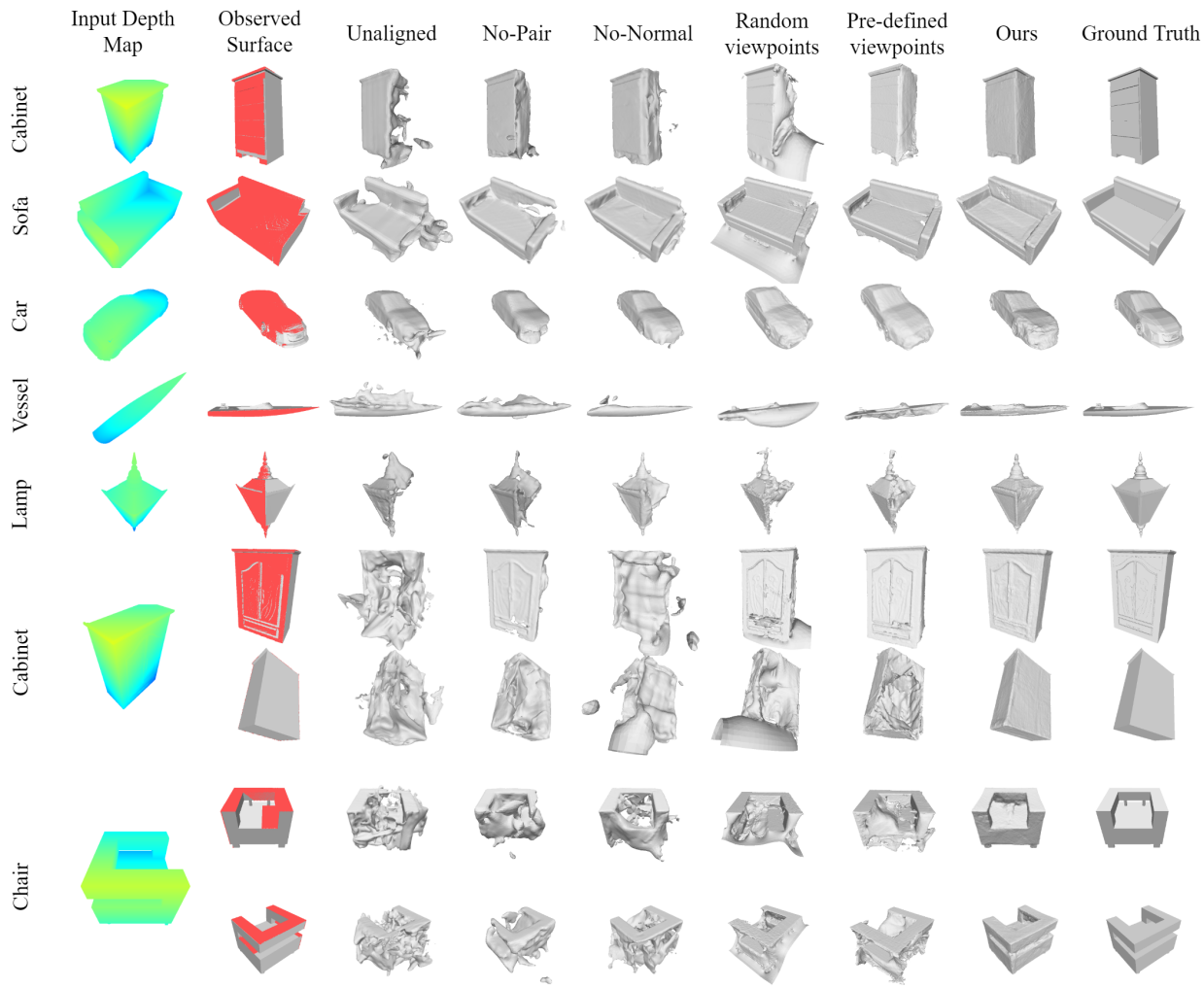


Figure 7. Qualitative comparison of the ablation study: forgoing to align the shape to the symmetry plane, inpainting without providing the symmetric counter-part, omitting the inpainted normals in the depth inpainting module, and replacing RL-agent with random viewpoints and pre-defined viewpoints.

References

[1] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: measuring error on simplified surfaces. In *Computer Graphics Forum*, volume 17, pages 167–174, 1998. 1

[2] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Eur. Conf. Comput. Vis.*, pages 85–100, 2018. 2

[3] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. Morphing and sampling network for dense point cloud completion. In *AAAI*, volume 34, pages 11596–11603, 2020. 1

[4] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 652–660, 2017. 1

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241, 2015. 2

[6] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 1

[7] Junshu Tang, Zhijun Gong, Ran Yi, Yuan Xie, and Lizhuang Ma. Lake-net: Topology-aware point cloud completion by localizing aligned keypoints. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1726–1735, 2022. 1