Multiresolution Deep Implicit Functions for 3D Shape Representation

Zhang Chen[1,2,*] Yinda Zhang[1] Kyle Genova[1] Sean Fanello[1] Sofien Bouaziz[1]
Christian Häne[1] Ruofei Du[1] Cem Keskin[1] Thomas Funkhouser[1] Danhang Tang[1]
[1] Google [2] ShanghaiTech University

Abstract

We introduce Multiresolution Deep Implicit Functions (MDIF), a hierarchical representation that can recover fine geometry detail, while being able to perform global operations such as shape completion. Our model represents a complex 3D shape with a hierarchy of latent grids, which can be decoded into different levels of detail and also achieve better accuracy. For shape completion, we propose latent grid dropout to simulate partial data in the latent space and therefore defer the completing functionality to the decoder side. This along with our multires design significantly improves the shape completion quality under decoder-only latent optimization. To the best of our knowledge, MDIF is the first deep implicit function model that can at the same time (1) represent different levels of detail and allow progressive decoding; (2) support both encoder-decoder inference and decoder-only latent optimization, and fulfill multiple applications; (3) perform detailed decoder-only shape completion. Experiments demonstrate its superior performance against prior art in various 3D reconstruction tasks.

1. Introduction

In recent years, deep implicit functions (DIF) have gained much popularity as a 3D shape representation in applications such as compression [31], shape completion [8], neural rendering [25, 34], and super-resolution [4]. In contrast to explicit representations such as point clouds, voxels, or meshes, a 3D shape is encoded into a compact latent vector, which when combined with a sampled 3D location as input to a decoder can be used to evaluate an implicit function for surface reconstruction.

In this paper, our objective is to design a DIF for shape representation that has three main properties: ① represent shapes with arbitrarily fine details (adding more bits to the representation provides more details), ② support both encoder-decoder inference and decoder-only latent optimization, and can be applied to different tasks, and ③ enable detail-preserving shape completion from inputs with large unobserved regions. These properties are all important for a shape representation. Yet, to the best of our knowledge, no prior method has achieved all three properties.

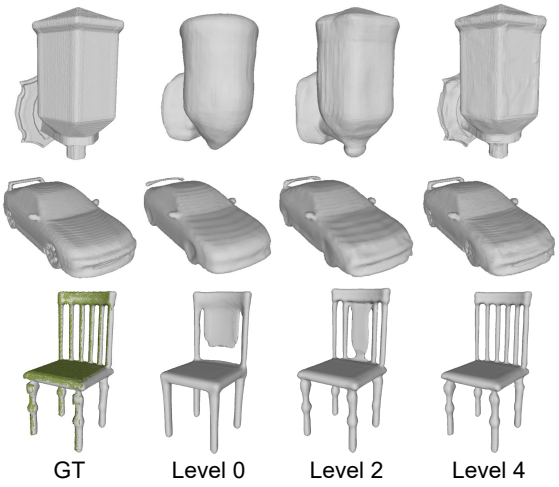


Figure 1: Example results of our model for auto-encoding (row 1 and 2) and shape completion (row 3) in different levels of detail. Green dots represent the observed depth pixels for the completion task.

Existing DIF methods can be classified into global and local approaches. Early methods mostly belong to the global category [27, 3, 23, 39, 24], where a single latent vector is used to represent the whole shape. These approaches learn to encode a global shape prior in a compact latent space, which can then be leveraged to fulfill various reconstruction tasks. However, due to the limited capacity of the latent space and the global nature of these approaches, global methods usually lack fine-grained detail.

More recently, local approaches [19, 1] have been proposed. These methods divide the space into local regions and encode each one with a latent vector. Such local representations provide better accuracy and generalization when representing shapes, especially under decoder-only latent optimization. However, they do not model a global prior. As a result, they cannot be used for shape completion with large unobserved regions since in such regions there is no data to optimize the latent vectors. To overcome this issue, [13, 4]

*Work done while the author was an intern at Google.

use an encoder to regress local latent vectors from incomplete inputs. However, their methods are limited to encoder-decoder inference when doing shape completion. Compared to decoder-only latent optimization, encoder-decoder inference has less flexibility on the inputs and is less accurate for preserving detail in observed regions.

In this paper, we propose a novel 3D representation: Multiresolution Deep Implicit Function (MDIF). The core idea is to represent a shape as a multiresolution hierarchy of latent vectors, where each level encodes different frequencies of an implicit function. The higher levels of our representation provide the global shape and the lower levels provide fine detail. Different from local methods [13, 4], MDIF has a *one-decoder-per-level* architecture, where each decoder produces a residual with respect to its parent level, like a Haar wavelet [6]. This simplifies learning of fine detail and enables progressive decoding to achieve arbitrary levels of detail (see Figure 1).

To enable detailed shape completion with decoder-only latent optimization, we further propose to use global connection across levels as well as applying dropout on the latent codes. The global connection serves to integrate global priors into lower levels to compensate for missing observations. Meanwhile, applying dropout on the latent codes simulates partial observation in the latent space during training, and therefore forces the decoders to learn to complete shapes under encoder-less scenario.

Overall, our model has the following merits:

1. Can represent complex shapes with high accuracy, and allows progressive decoding for different levels of detail.

2. Supports both encoder-decoder inference and decoder-only latent optimization, and is effective for different applications as illustrated in the experimental results.

3. Enables detailed decoder-only shape completion that accurately preserves detail in observed regions while producing plausible results in unobserved regions.

2. Related Work

There are largely two types of 3D geometry representations in computer graphics and vision. *Explicit representations* such as meshes, splines, and point clouds, are widely adopted in the field of CAD and real-time rendering [10, 11], since they are compact and highly optimized for editing and rendering. *Implicit representations*, such as the zero-level set of a signed distance field, have gained increasing popularity in volumetric capture [9, 18, 26, 10, 7], since they can represent arbitrary surface topology and define watertight surfaces.

Convolutional neural networks (CNNs) have been proposed for predicting an implicit representation of objects. Early techniques were only able to predict low-resolution grids [15, 5, 38]. More recently, methods relying on an octree structure have been proposed [32, 16, 28, 36, 37] to avoid the cubic growth inherent to high-resolution grids. However, the implicit representation learnt by these networks is still discrete, potentially creating discretization artefacts when reconstructing 3D shapes. To overcome this limitation and allow for learning the implicit representation over the continuous domain, the problem can be reformulated as a multi-layer perceptron (MLP) which takes the location at which the implicit representation is to be evaluated as input [27, 3, 23, 39, 24]. This allows for querying the implicit representation at continuous locations during test time. Termed as *Deep Implicit Functions* (DIF), this technique can be categorized into global, local, and hierarchical methods.

Global methods. Global methods represent a 3D shape with a single holistic latent code. The projection to the latent space can be done via an encoder [3], or latent optimization [27]. A decoder is then used to recover the shape from the latent vector. To obtain a smooth manifold on the latent space for shape generation, people have developed optimization strategies based on auto-decoding [27], curriculum learning [12], and adversarial training [21]. Global methods are robust to local noise, hence have good shape completion capability. However, these approaches have difficulty recovering fine detail. Recent methods [29, 30, 25] propose to use periodic activation functions to lift the input positional vector to high dimensional space allowing to better preserve high frequency detail. However, these methods focus on per-instance fitting instead of generalization to new scenes and objects.

Local methods. In contrast, local methods uniformly divide the 3D space into local grids [19, 1, 4] or use an encoder to decompose space into local parts [14, 13]. Then they either assign each local grid/part with a latent code [19, 1, 13] or trilinearly interpolate feature grids to obtain the latent code at each querying location [4]. Since each latent code only needs to represent the shape in a local region, it is much easier to encode detail and generalize to unseen objects. However, these methods do not include global context, hence it is not feasible to perform decoder-only shape completion when there are large unobserved regions. While the feature grids used by [4] span multiple resolutions, they still do not contain global context and are only used to represent single level of detail.

Hierarchical methods. Some methods perform shape reconstruction in stages, where a low-resolution shape prediction precedes a high-resolution prediction [8, 20, 17, 35]. For example, Global-To-Local Generative Models [35] decode a global voxel grid and then add detail with a part-wise refiner. NSVF [22] uses an octree hierarchy of implicit functions to represent the radiance field for neural rendering. Though

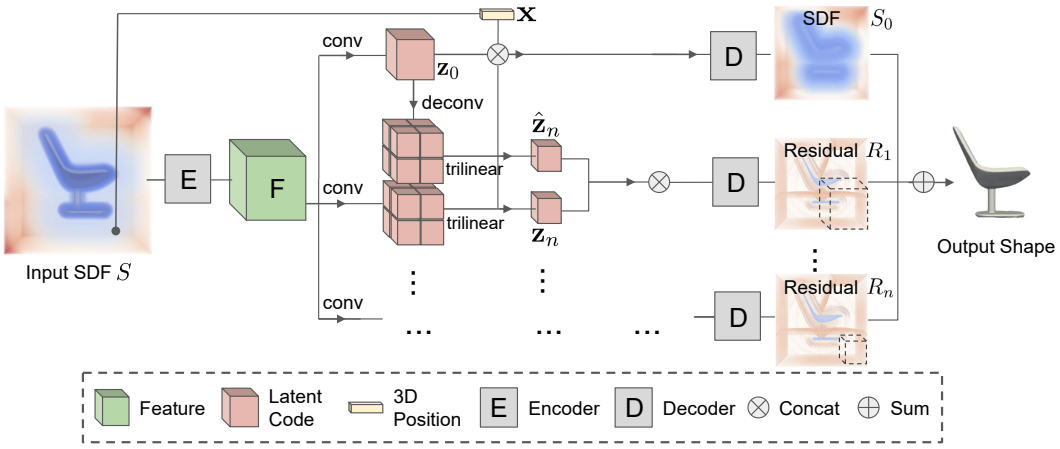


Figure 2: Starting from the input SDF S, we first extract a global feature F, which is then encoded into different levels of latent codes through 3D convolutions and transposed convolutions. The decoder is performed per-level to support different resolutions. The outputs of the pipeline consist of a global SDF S_0 and multiple residuals R_n at different scales, which are used to compute the final reconstruction.

their motivation for using an octree is similar as ours, they do not use it to represent a globally consistent shape, but rather a view-dependent radiance function suitable for view synthesis. They would not be able, for example, to perform shape completion.

3. Methodology

Our overarching goal is to design a flexible representation that can generate shapes from coarse to fine resolutions for reconstruction or completion tasks. Depending on the application, our model can perform inference in the encoder-decoder mode for efficiency or the decoder-only latent optimization for better accuracy. To achieve this, our pipeline, shown in Figure 2, encodes the input SDF into multiple levels of latent codes. Each level has a decoder reconstructing in a different detail level. To detail our design, we first formulate a multi-resolution representation in the form of traditional implicit function in Section 3.1. Then in Section 3.2, we explain how to design a deep neural network version of this representation. The training process is described in Section 3.3. Finally in Section 3.4, we explain different inference modes with respect to different applications.

3.1. Multires Implicit Function

We choose to learn the signed distance function (SDF), which is a level set defined as:

$$V(\tau) = \{\mathbf{x} : S(\mathbf{x}) = \tau\} \tag{1}$$

where V is the volume containing the shape, $\mathbf{x}$ is a 3D point inside V, and $S : \mathbb{R}^3 \to \mathbb{R}$ is the SDF function that represents the signed distance to the closest surface (positive on the outside and negative on the inside). We then use $S(\mathbf{x})$ to represent the SDF value of a particular point $\mathbf{x}$, and $V(0)$ to represent the surface or zero-crossing.

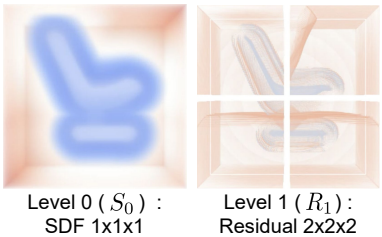


Figure 3: Octree subdivision and decoded outputs of the first two levels: (left) level 0 contains a single cell and decodes into SDF; (right) level 1 contains a 2^3 grid and decodes into residuals. Aggregating all levels we have the final SDF with fine details.

Now we can define an N-level version of S as $\{S_n\}$, $n = 0 \dots N-1$, where each level represents different frequency of details from low to high. To construct this, we subdivide V into an N-level octree. Unlike conventional octrees which only subdivide non-empty cells, our tree is balanced because completing partial observation is one of our target scenarios.

For level 0 (the coarsest level), geometry is represented as SDF S_0; for level $n > 0$, we use the residual $R_n = S_n - S_{n-1}$ to capture finer details, as shown in Figure 3. The final SDF reconstruction is therefore defined as $S = S_0 + \sum_{n=1}^{N-1} R_n$. In Section 4.2, we empirically show that inferring residuals yields better performance compared to directly regressing the SDF.

3.2. Multires Deep Implicit Function

The idea of a *deep* version of the multires implicit function, is to encode the shape in each cell of the octree into a latent code $\mathbf{z}$ with DNN. For a cell in level $n = 0$, its latent code represents an SDF; while for a cell in $n > 0$, its latent code encodes residuals. Eventually we end up having a tree of latent codes Z, where the latent codes in each cell of level n form a latent grid Z_n at this level. The spatial resolution

and total capacity of the latent grids increase with the level, and consequently the level of detail gets higher.

In Figure 2, we describe the design of our network architecture to encode the shape into Z. On the encoder side, the input is the regular grid form of a SDF S with a resolution of 128^3. The encoder $\mathcal{E}$ first extracts a global feature F from S. Then F is encoded into different levels of latent grids through 3D convolution layers. Note that at level 0, there is only one latent code representing the global shape which is critical for completion tasks.

On the decoder side, unlike [4], our model has one decoder per level to support different levels of detail. For the decoder module at each level, we choose IM-Net [3] which consists of several fully-connected layers. The remaining question is *what do we input to the decoders?* At the global level ($n = 0$), since there is only one latent code, the decoder $\mathcal{D}_0$ simply takes $\mathbf{z}_0$ and a 3D position $\mathbf{x}$ as input, and decodes the SDF value at that point. For higher levels ($n > 0$), the input of decoder $\mathcal{D}_n$ consists of two parts. The first part is similar to [4], we use trilinear interpolation to sample a latent code $\mathbf{z}_n$ from the latent grid of this level as $Z_n(\mathbf{x})$, based on the 3D location $\mathbf{x}$. For the second part, we first apply deconvolution to upsample $\mathbf{z}_0$ to a latent grid $\hat{Z}_n$, which has the same spatial resolution as Z_n. Then trilinear interpolation is also applied to sample a latent code $\hat{\mathbf{z}}_n$ from $\hat{Z}_n$. This allows the decoder $\mathcal{D}_n$ to have access to the global context to better decode local details as well as compensating for missing data during shape completion. We call this *global connection*. Formally,

$$\begin{aligned} \mathcal{D}_0(\mathbf{z}_0, \mathbf{x}) &= S_0, \\ \mathcal{D}_n(\mathbf{z}_n, \hat{\mathbf{z}}_n) &= R_n, n > 0. \end{aligned} \tag{2}$$

Note that for $n > 0$, the decoders do not need to take 3D positions $\mathbf{x}$ as input, because $\mathbf{z}_n$ and $\hat{\mathbf{z}}_n$ are already functions of $\mathbf{x}$ via trilinear interpolation. Finally, since $\mathcal{D}_{n>0}$ predicts residual R, the outputs of all levels are aggregated to have the final SDF. For detailed network architecture, please refer to our supplementary.

3.3. Training

MDIF is trained end-to-end in an encoder-decoder fashion because: ① it allows both encoder-decoder inference and decoder-only latent optimization to be available during test-time; ② training with an encoder is generally more efficient comparing to training in decoder-only mode, since latent codes are not initialized randomly.

Points Sampling. We generate 128^3 regular SDF grids as the input of the encoder $\mathcal{E}$. In addition, the decoders require a 3D point set as training data. Similar to [13], we sample a uniform point set $\mathcal{P}_U$ inside the object bounding box, as well as a near-surface point set $\mathcal{P}_S \subset \{(\mathbf{x}, S(\mathbf{x})) : |S(\mathbf{x})| < 0.04\}$ for each training object. Each point set has 100K samples. Mixing the two gives us the final training set $\mathcal{P} = \mathcal{P}_U \cup \mathcal{P}_S$, which implicitly applies more weight to the near-surface points. At each training iteration, 4096 samples are randomly drawn from each set.

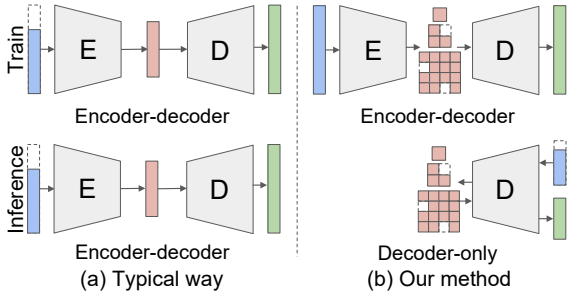


Figure 4: (a) The conventional way of training an auto-encoder for completion is to feed partial data (blue) from the encoder side. In this way, the encoder plays a crucial role in completion during inference. (b) We instead apply random dropout to our latent grids during training (top), which forces the decoder to learn to complete the shape (green). As a result, detailed completion can be achieved with decoder-only latent optimization (bottom). For simplicity, we visualize levels of decoders as one block.

Loss. During training, our final loss is the summation of losses at all levels, such that $\mathcal{L} = \sum_{n=0}^{N} \mathcal{L}_n$. For each level n, we first aggregate the predicted SDF and residual up to this level to produce S_n, and then measures the L1 difference between it and groundtruth $\bar{S}$. Formally,

$$\mathcal{L}_n = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{x} \in \mathcal{P}} \left| S_n(\mathbf{x}) - \bar{S}(\mathbf{x}) \right|. \tag{3}$$

Latent grid dropout. There are mainly two standard ways to make a deep implicit function model work for completion tasks. The more conventional way, as illustrated in Figure 4 (a), is training the model to take partial data as input and complete them. In this manner, the completion functionality is distributed among the encoder and decoder, therefore different encoders need to be trained for different completion tasks. Another way is decoder-only latent optimization, where the encoder is not needed during test-time and the latent code is optimized based on partial data [27]. This manner provides higher accuracy on observed regions and directly generalizes to different completion modalities (depth image, partial scan, etc.) without retraining. However, it only works for global methods and cannot be applied to local methods. The reason is that for unobserved regions with no data point, the corresponding local latent codes cannot be optimized and will stay as initialization. Such latent codes would then be decoded into wrong shapes by the decoder.

To address this, we propose to train with complete shapes, but apply random dropout to latent grids, as shown in Figure 4 (b). The motivation is to simulate partial data in the latent space rather than the input space, hence forcing the

decoder to learn to complete shapes without encoder. Specifically, for each level $n > 0$, we apply spatial dropout to Z_n, but keep the full content of $\hat{Z}_n$, so that the decoder can utilize the global context from level 0. Note that our proposed multi-level architecture and *global connection* make this dropout strategy possible during training: this cannot be applied to other global or local approaches, without substantial changes in the architectures.

3.4. Inference

We discuss our inference process with respect to auto-encoding (complete observation) and shape completion (partial observation).

Auto-encoding. MDIF supports both encoder-decoder inference and decoder-only latent optimization. For applications that emphasize efficiency, encoder-decoder inference is a better choice, as it only has one feed-forward pass. For applications that require accuracy, decoder-only latent optimization is preferred.

Shape completion. Here we focus on shape completion from a single depth image via decoder-only latent optimization, due to its benefits in accuracy and generalizability. We initialize all latent codes as zeros. Similar to global methods, level 0 can be optimized to have a coarse but complete reconstruction. For higher levels, the decoder is trained to add detail onto the observed parts, while produce sparse residual to the unobserved part. For this optimization process to work, we need to properly sample points and modify the loss function to accommodate incomplete observation.

When sampling the point set $\mathcal{P}$ from a depth image, since part of the shape is occluded, we cannot simply sample points in the full volume as in training. Instead, we apply raycasting to sample camera-observable points as $\mathcal{P}_V$, and occluded points as $\mathcal{P}_O$. For level $n = 0$, the loss function is the same as Equation 3 except only applied to visible points $\mathcal{P}_V$. For level $n > 0$, the loss function $\mathcal{L}_n$ is modified to contain two terms as follows:

$$\begin{aligned} \mathcal{L}_n &= \mathcal{L}_n^V + \lambda \mathcal{L}_n^O, \\ \mathcal{L}_n^V &= \frac{1}{|\mathcal{P}_V|} \sum_{\mathbf{x} \in \mathcal{P}_V} \left| S_n(\mathbf{x}) - \bar{S}_n(\mathbf{x}) \right|, \\ \mathcal{L}_n^O &= \frac{1}{|\mathcal{P}_O|} \sum_{\mathbf{x} \in \mathcal{P}_O} \left(1 - G_\sigma \left(d(\mathbf{x}, \mathcal{P}_V)\right)\right) \left| R_n(\mathbf{x}) \right|. \end{aligned} \tag{4}$$

The first term $\mathcal{L}_n^V$ is to minimize the difference between aggregated SDF prediction and ground truth for visible points. The second term $\mathcal{L}_n^O$ is for regularizing the residual of occluded points, such that the global shape from level 0 will be preserved for the unobserved part. In particular, $d(\mathbf{x}, \mathcal{P}_V)$ measures the closest distance from an occluded point $\mathbf{x}$ to the visible point set $\mathcal{P}_V$, and is normalized by a Gaussian G of standard deviation σ. In practice, we empirically set $\lambda = 10$ and $\sigma = 0.1$. We call the second term *global consistency*.

4. Experiments

In this section, we first validate the benefits of our proposed components by ablating important aspects (Section 4.2). Then to evaluate the effectiveness of our approach, we compare with state-of-the-art methods on auto-encoding 3D shapes (Section 4.3) and applications including point cloud completion (Section 4.4), voxel super-resolution (Section 4.5) and shape completion from depth image (Section 4.6). These experiments demonstrate the capability of our method under different tasks and inference modes. We use 5 levels for MDIF in the experiments and set the dimensions of the latent grids $\{Z_n\}$ as: $[1^3 \times 512, 2^3 \times 64, 4^3 \times 32, 8^3 \times 16, 16^3 \times 8]$. But note that MDIF is flexible to use any number of levels. During decoder-only latent optimization, we fix all other network parameters and only optimize over $\{Z_n\}$. Please refer to supplementary for more implementation details.

4.1. Dataset & Metrics

Following prior works [19, 13], we run the experiments on the ShapeNet dataset [2] with train/test splits from 3D-R^{2}N^2 [5], which contain a subset of 13 categories in ShapeNet. We use all 13 categories in our experiments except for ablation studies where we only use the chair category. In all experiments, we only take the train split for training and leave out the test split for evaluation. For metrics, we use the *Chamfer L2 distance* and *F-Score* with the exact settings as in [13]. Since the Chamfer distance measures the average errors of all points, while the F-Score measures the ratio of good predictions, these two metrics do not always agree with each other: a better F-Score with a higher Chamfer distance usually indicates a few outliers resulting in significant error.

4.2. Ablation Study

We conduct our ablation studies on the chair category of ShapeNet, for it contains large number of instances as well as significant intra-class shape variance. The models are all trained under encoder-decoder scheme and use decoder-only latent optimization during inference.

Global/local/hierarchical. We compare MDIF with a global and a local baselines to emphasize the impact of MDIF 's hierarchical model. The global baseline only has level 0, whilst the local baseline has only level 4 (a 16^3 latent grid). In Table 1, we compare with the baselines in terms of auto-encoding and shape completion from depth image. For auto-encoding, the local baseline clearly outperforms global, since it has larger capacity and the capability to capture details. On the flip side, for shape completion, the global baseline has better accuracy because the local baseline behaves randomly on the unobserved part, as visualized in the column 3 of Figure 5. Our MDIF however, incorporates the benefits of global and local levels, and produces superior

Method	Auto-encoding		Shape Completion	
	Chamfer (↓)	F-Score (↑)	Chamfer (↓)	F-Score (↑)
Ours	**0.009**	**99.5**	**1.34**	**66.5**
Global baseline	0.228	88.7	1.56	63.7
Local baseline	0.012	99.2	5.47	48.3

Table 1: **Quantitative comparisons among global/local/hierarchical baselines.** The local baseline has better auto-encoding performance than global, but performs poorly for shape completion from depth image. Our method combines the benefits of both.

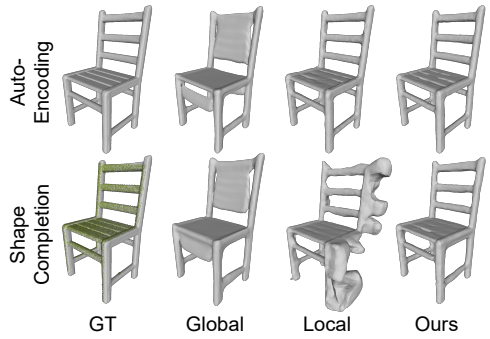


Figure 5: **Qualitative comparisons among global/local/hierarchical baselines.** The global baseline lacks detail but behaves reasonably in both applications. The local method works well on observed data (green dots) but generates noisy shapes for unobserved part. Our method has superior performance in both scenarios.

results in both tasks.

Network components. In Table 2, we incrementally compare the impact of four network components during decoder-only latent optimization.

Global consistency loss (Equation 4), which is designed to work for shape completion, has marginal improvements on the overall completion numbers. However, the column 3 of Figure 6 shows that it is still important for clean reconstruction in unobserved regions.

We also compare the difference between decoding into SDF S or *residual* R in Equation 2. Since predicting residual forces lower levels to focus on the addition of fine detail, it is a stronger constraint and improves both auto-encoding and shape completion.

Latent grid dropout is another component that is tailored to shape completion. Without it, the Chamfer error drastically increases from 3.0 to 8.38. Also, it slightly improves decoder-only auto-encoding. We hypothesize it is because dropout improves the generalization of the decoders at levels 1-4 to test data and reduces the ambiguity between levels.

Finally, *global connection* passes the global shape prior to other levels. Without it, the completion results are almost unconstrained on the unobserved part. It also helps auto-encoding, since without it, we are asking the network to add more detail without knowing what has been predicted by the previous levels, which is not sensible.

Method	Auto-encoding		Shape Completion	
	Chamfer (↓)	F-Score (↑)	Chamfer (↓)	F-Score (↑)
Full pipeline	**0.009**	**99.5**	**1.34**	**66.5**
No consistency loss	-	-	1.43	64.7
No residual	0.025	98.2	3.00	53.0
No dropout	0.026	97.9	8.38	43.0
No global connection	0.086	93.9	19.9	39.6

Table 2: **Quantitatively ablate the impacts of different components on auto-encoding and shape completion from depth image.**

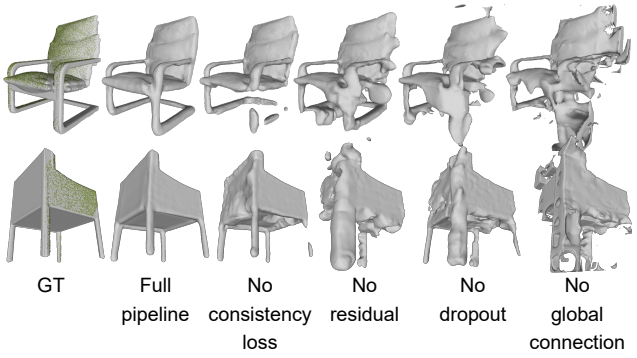


Figure 6: **Qualitative ablation of the impacts of different components on shape completion from depth image.** Green dots are projected depth pixels (observed data). Note that every component is necessary for good results.

4.3. Auto-Encoding 3D Shapes

Accuracy on test split. We first evaluate the auto-encoding accuracy under encoder-decoder inference for the test shapes in 3D-R^{2}N^2. We compare our approach with state-of-the-art DIF methods including OccNet ("Occ.") [23], SIF [14], LDIF [13] and IF-Net ("IF.") [4]. The results for OccNet, SIF and LDIF are kindly provided by the authors of [13]. For IF-Net, it originally uses high-resolution latent grids (up to 128^3) which altogether is over 20 times larger than the input grid (128^3) in the number of parameters. This would make the encoded latent grids meaningless for auto-encoding task. Therefore in this experiment, we constrain IF-Net to only use latent grids up to 16^3 resolution (same as our approach) and have same total number of parameters in the latent grids as our approach. Table 3 (middle columns) show the average metrics across 13 categories. Our method achieves slightly higher F-Score and much lower Chamfer error, which means it works better overall and on hard examples too. As visualized in Figure 7, our method preserves details well and represents thin structures much better than the competing methods (see the last row).

Next, we evaluate the performance under decoder-only latent optimization. We compare with OccNet ("Occ.") [23], IM-Net ("IM.") [3] and a local baseline (resembles [19, 1]), as shown in Table 3 (right columns). Our method also performs the best under this inference mode and can improve

	Occ.	SIF	LDIF	IF.	Ours	Occ.*	IM.*	Local*	Ours*
Chamfer	0.49	1.18	0.4	0.39	**0.19**	0.43	0.46	0.14	**0.10**
F-Score	81.9	59	92.2	92.9	**93.0**	81.4	86.7	96.9	**97.0**

Table 3: **Auto-encoding accuracy for objects in 3D-R^{2}N^2 test set.** Middle columns compare methods under encoder-decoder inference while right columns compare under decoder-only latent optimization. $*$: decoder-only latent optimization.

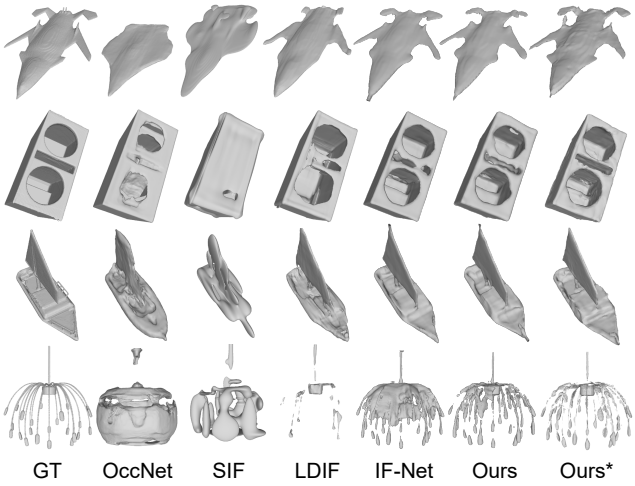


Figure 7: **Auto-encoding results on test split.** Our method better reconstructs the groundtruth and recovers fine details. $*$: decoder-only latent optimization.

over encoder-decoder inference by a large margin. The last column of Figure 7 shows qualitative results.

Generalizability. In this experiment, we study the generalizability to shapes vastly different from training data. We test the trained models from the last experiment without fine-tuning on 10 ShapeNet categories that are unseen during training. In Table 4, we compare the performance under both inference modes, and our method respectively outperforms other methods. While global methods generalize poorly to unseen categories, our method performs equally well as seen categories. Qualitative results are shown in Figure 8.

Progressive refinement. One unique property of MDIF is the capability to decode shapes in different levels of detail. This enables the progressive refinement application in graphics, where 3D data are encoded into different levels of detail and progressively rendered. Since MDIF has a multi-level architecture, this can be easily achieved by only decoding the shape up until a certain level. Figure 9 shows the distortion against the accumulated latent code size in bytes of each level, *i.e.*, latent space capacity. MDIF consistently improves with each level added. When under similar bytes, MDIF still outperforms SIF, LDIF and IF-Net.

4.4. Point Cloud Completion

In this application, we take voxelized point cloud instead of SDF grid as input. We follow the same steps as IF-Net [4]

	Occ.	SIF	LDIF	IF.	Ours	Occ.*	IM.*	Local*	Ours*
Chamfer	0.85	1.48	0.53	0.40	**0.17**	0.62	0.47	0.063	**0.054**
F-Score	66.6	43.0	84.4	92.4	**92.8**	71.1	80.5	97.5	**97.5**

Table 4: **Auto-encoding accuracy for objects in unseen categories.** Middle columns compare methods under encoder-decoder inference while right columns compare under decoder-only latent optimization. $*$: decoder-only latent optimization.

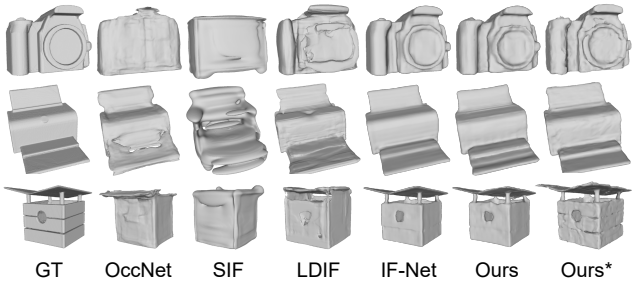


Figure 8: **Auto-encoding results for unseen categories.** $*$: decoder-only latent optimization.

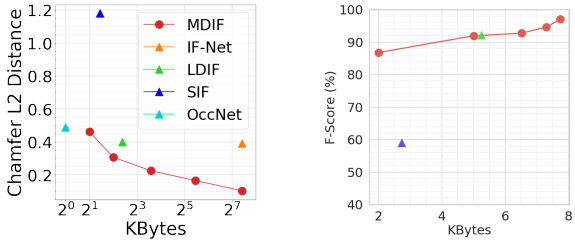


Figure 9: **Progressive refinement rate-distortion.** Our model allows progressive transmission of the latent codes of each level for refinement. This figure shows the accumulated latent code size (in bytes) and the respective distortion. For reference, the original 128^3 volume is 8MB.

to produce such input: first sample 300 points from object surface and then voxelize these points into a 128^3 grid. We compare our method with IF-Net, where both methods use encoder-decoder inference. As indicated in Table 5 (middle 2 columns), our method has higher F-Score and much lower Chamfer error. This reveals that our method is more accurate and stable in prediction. Figure 10 (top row) shows results for one example data. Our method preserves the cavity in the legs while IF-Net incorrectly fills part of the cavity.

4.5. Voxel Super-Resolution

In this task, we input 32^3 occupancy grid and ask the network to predict the underlying continuous implicit field. The resolution of output grid for meshing is 128. We compare our method with IF-Net, with both under encoder-decoder inference. Table 5 (right 2 columns) show the quantitative results. Similar to the case in point cloud completion, our method outperforms IF-Net with a large margin in Chamfer error. In Figure 10 (bottom row), we show qualitative results

Method	Point Cloud Completion		Voxel Super-Resolution	
	Chamfer (↓)	F-Score (↑)	Chamfer (↓)	F-Score (↑)
IF-Net	1.61	85.0	1.82	65.4
Ours	**0.39**	**86.1**	**0.96**	**66.9**

Table 5: Quantitative results for point cloud completion and voxel super-resolution.

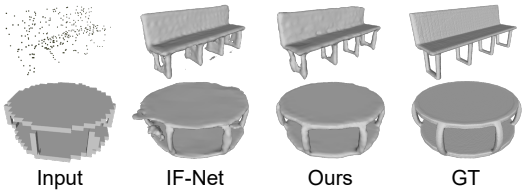


Figure 10: Qualitative results for point cloud completion (top row) and voxel super-resolution (bottom row). More qualitative results are available in supplementary.

on one example data. Our method is reasonably accurate in both global shape and local detail while IF-Net produces artifacts near the object boundary.

4.6. Shape Completion from Depth Image

Our final experiment investigates shape completion from depth image. We compare MDIF with IM-Net, OccNet and LDIF. OccNet and LDIF use encoder-decoder inference while IM-Net and MDIF use decoder-only latent optimization. Note that for IM-Net and MDIF , we directly use the model trained in the auto-encoding task (Section 4.3) without retraining or finetuning. This is considered a benefit of decoder-only latent optimization. Figure 11 reports the percentages of surface points with distance to groundtruth smaller than different thresholds. MDIF has a good proportion of points with low error and consistently outperforms IM-Net at all thresholds, reflecting its advantage on preserving details in observed regions. However, MDIF has higher error in unobserved regions than methods under encoder-decoder inference (OccNet, LDIF). This is illustrated in Figure 12, where the errors of our results are mostly on the occluded side. For example, in row 4 where the table top is completely unobserved, our estimation is thicker than groundtruth, hence resulting in higher error. Despite this, the predicted shape still looks plausible. This and other examples suggest that the Chamfer distance and F-Score are suited for assessing the observed parts, but not for the unobserved parts where many plausible solutions exist. Therefore, to evaluate plausibility, we further conduct a user study that votes between MDIF and LDIF results on 32 pairs of examples (please refer to supplementary for details). The results show that 54.2% of the participants chose MDIF results as more plausible, whilst 31.9% thought LDIF results were better. In addition, 13.9% could not decide between MDIF and LDIF. Moreover, when compared with the quantitative metrics, 68.1% disagree with the Chamfer distance, and 51.4% disagree with the F-Score.

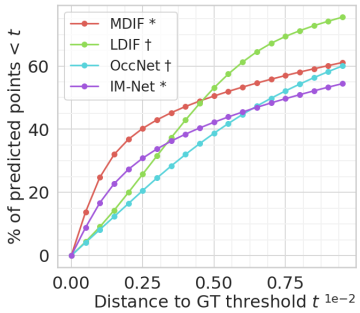


Figure 11: **Shape completion from depth image.** The proportion of predicted points with distance to groundtruth smaller than different thresholds. †: encoder-decoder inference; ∗: decoder-only latent optimization.

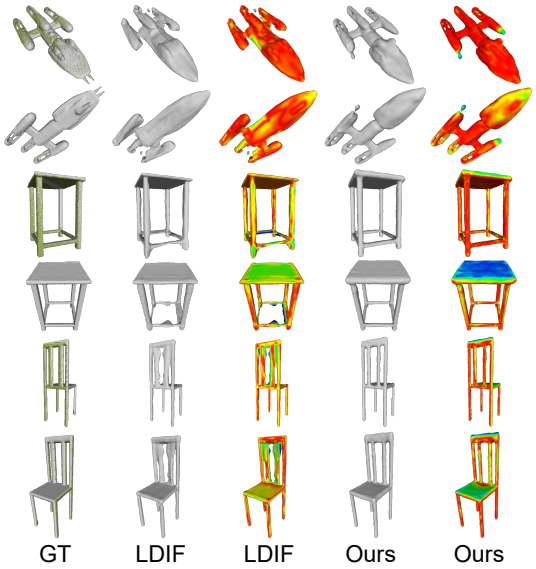


Figure 12: **Qualitative results for shape completion from depth image.** We visualize the reconstruction and error maps (low/mid/high) of three objects from two different angles. In GT column, green dots represent observed parts.

5. Conclusion

In this paper, we present MDIF, a multi-resolution deep implicit function to progressively represent and reconstruct geometries. MDIF is trained end-to-end in an encoder-decoder fashion and supports both encoder-decoder inference and decoder-only latent optimization. We demonstrate that MDIF outperforms state-of-the-art methods on tasks including auto-encoding 3D shapes, point cloud completion and voxel super-resolution. We further show that MDIF enables detailed decoder-only shape completion from a depth image: the details in observed regions are accurately preserved while the unobserved regions are completed with plausible shapes. In the future, we would like to explore transferring details from observable parts to occluded parts in completion tasks. We also plan to apply MDIF to more applications such as shape manipulation.

Multiresolution Deep Implicit Functions for 3D Shape Representation (Supplementary Material)

Zhang Chen[1,2,*] Yinda Zhang[1] Kyle Genova[1] Sean Fanello[1] Sofien Bouaziz[1]
Christian Häne[1] Ruofei Du[1] Cem Keskin[1] Thomas Funkhouser[1] Danhang Tang[1]
[1] Google [2] ShanghaiTech University

6. Supplementary Material

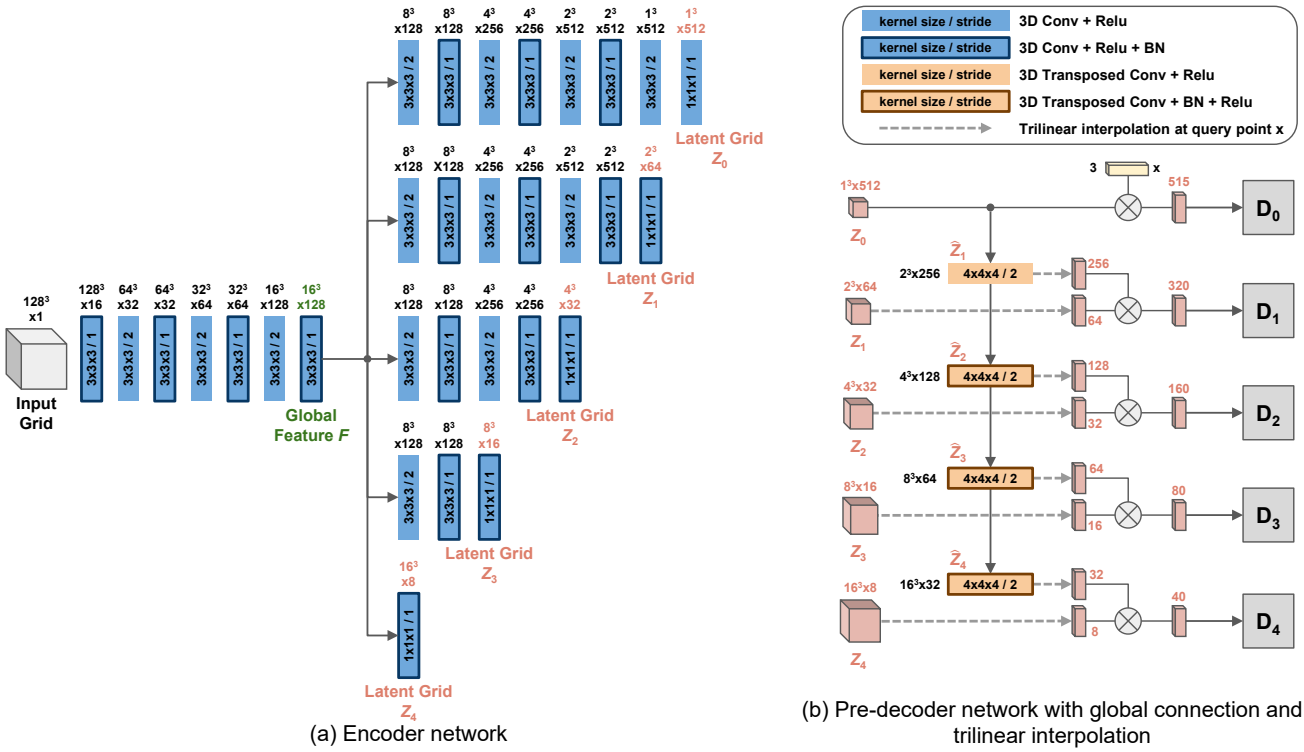
6.1. Implementation Details

Detailed network architecture. Figure 13 shows the detailed architecture of our network. On the left, Figure 13 (a) is the encoder network that is used in training and encoder-decoder inference. It takes 3D grid as input and outputs the latent grid Z_n of each level. For the voxel super-resolution experiment (Section 4.5), since the input is only 32^3, we accordingly remove the first 4 convolution layers along with their activation and normalization layers.

On the right, Figure 13 (b) is the pre-decoder network. With latent grids $\{Z_n\}$ as input, it includes global connection and trilinear interpolation. The global connection consists of 3D transposed convolution layers to propagate global context from level 0 to other levels. Trilinear interpolation is utilized to obtain the latent codes at each query point, which are then fed into the decoders at each level. For level 0, the 3D position of query point is also fed into the decoder. For the decoder modules, we use the same IM-Net [3] architecture for each level, with the only difference in the input dimension.

*Work done while the author was an intern at Google.

*Work done while the author was an intern at Google.



(a) Encoder network

(b) Pre-decoder network with global connection and trilinear interpolation

Figure 13: Detailed architecture of our network.

Hyperparameters. We implement our method in TensorFlow. During training, we set batch size as 8 and train our network end-to-end. We use Adam as optimizer, with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a learning rate of $1e{-}4$. The latent grid dropout rate is set as 0.5 for the models that need to carry out decoder-only latent optimization while it is set as 0 for the models that only run encoder-decoder inference (*e.g.*, the models for point cloud completion and voxel super-resolution).

During decoder-only latent optimization, we optimize over $Z_n, n = 0, 1, ..., 4$ and keep other parameters fixed. We use Adam with the same configuration of β_1, β_2 as training, but at a higher learning rate of $1e{-}2$ to accelerate convergence. In all our experiments, we only run latent optimization for 1000 steps. For each step during auto-encoding, we randomly draw 2048 points. For each step during shape completion, we randomly draw 2048 camera-observable points, along with 1024 occluded points for the *global consistency loss*.

Experiment details. For the training data, we use the watertight ShapeNet meshes from OccNet [23] and normalize into bounding box with side length 1.28. We also truncate SDF values at 0.05.

For the auto-encoding experiment (Section 4.3), as mentioned in the paper, IF-Net [4] originally uses high-resolution latent grids which contain more parameters than the input grid. We therefore constrain IF-Net to only use latent grids with dimensions: $[8^3 \times 22, 16^3 \times 8]$. The resulting total number of parameters in the latent grids is the same as MDIF .

For the point cloud completion (Section 4.4) and voxel super-resolution (Section 4.5) experiments, unlike auto-encoding, the goal is to infer missing data rather than learn a compact latent space. Therefore, in these experiments, we use the original implementation of IF-Net which exploits high-resolution latent grids. Similarly, for MDIF in these experiments, we additionally interpolate features at query points from high-resolution feature grids and feed into the decoders.

6.2. Encoder-Decoder vs. Decoder-Only Inference

In Figure 14, we show qualitative auto-encoding results of MDIF using encoder-decoder inference and decoder-only latent optimization. Compared with encoder-decoder inference, decoder-only latent optimization already produces more accurate reconstruction with only 200 optimization steps. More steps further lower the error.

6.3. Illustration of Ablation Baselines

In Figure 15, we illustrate the baselines that we ablate in Table 1 and Table 2.

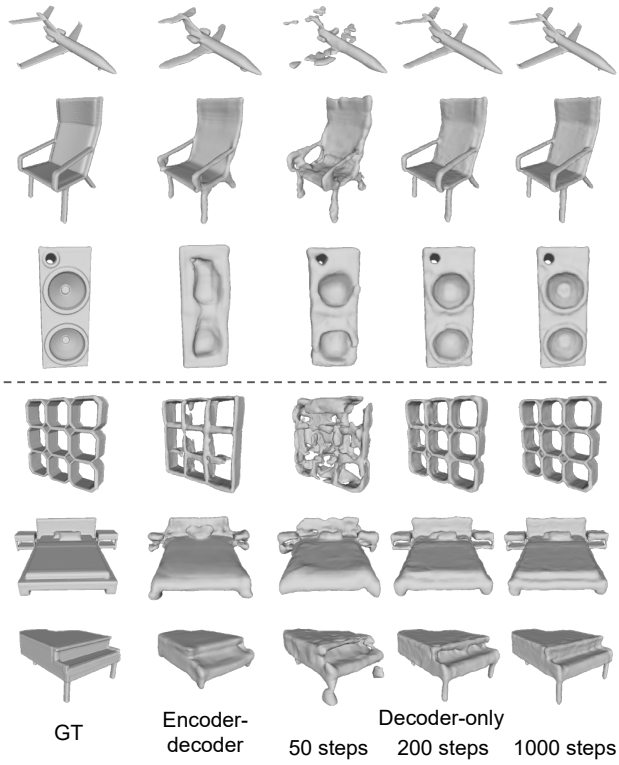


Figure 14: **Encoder-decoder vs. decoder-only inference.** Auto-encoding results of MDIF under encoder-decoder inference and decoder-only latent optimization. Top 3 rows: objects in 3D-R^{2}N^2 test split. Bottom 3 rows: objects in unseen categories.

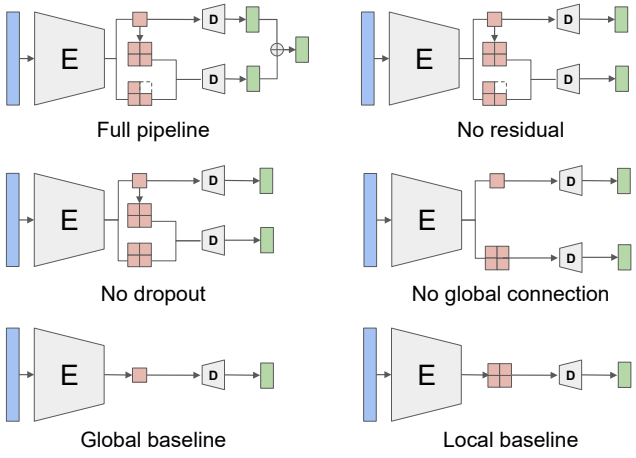


Figure 15: **Illustration of ablation study baselines.** E: encoder; D: decoder.

6.4. Comparison of Dropout and Consistency Loss

To further analyze the different contribution of *latent grid dropout* and *global consistency loss* on shape completion,

Method	Shape Completion Chamfer (↓)	F-Score (↑)
Full pipeline	**1.34**	**66.5**
No consistency loss	1.43	64.7
No dropout (leave-one-out)	1.43	63.9

Table 6: Quantitatively ablate the impacts of consistency loss and latent grid dropout on shape completion from depth image.

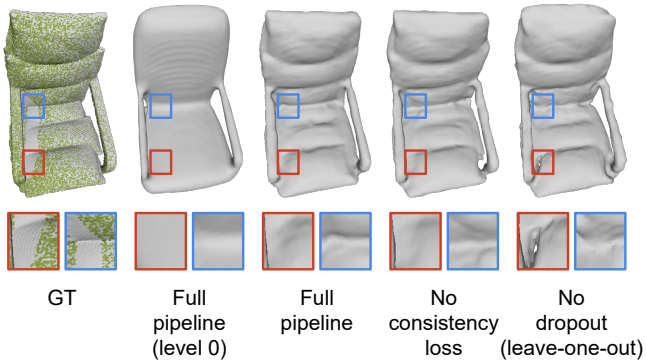


Figure 16: **Ablation on latent grid dropout and consistency loss for the task of shape completion.** Green dots are observed depth points. Compared to the global consistency loss which regularizes regions far from observed points, latent grid dropout reduces noisy residuals and enables plausible detail synthesis on regions that are close to the observed part.

we carry out a leave-one-out ablation on dropout where the only difference with full pipeline is the removal of latent grid dropout. Same as the baselines in Table 2, this ablation is conducted on the chair category of ShapeNet. In Table 6, we show that the removal of dropout leads to slightly larger decrease in quantitative performance than the removal of consistency loss. Meanwhile, dropout impacts qualitative results in a different way than the consistency loss. As shown in Figure 16, when dropout is applied (the third and fourth columns from the left), the model is able to synthesize plausible details on the unobserved regions that are close to the observed part (see insets at the bottom). On the contrary, without dropout (the rightmost column), the model tends to produce noisy residuals (red inset) or add no detail due to the consistency loss (blue inset).

6.5. Failure Cases

Figure 17 shows our failure cases under decoder-only latent optimization for auto-encoding and shape completion from depth image. For objects with very complex geometry or thin structures, our approach still faces challenges. For auto-encoding, such problems could be alleviated by using

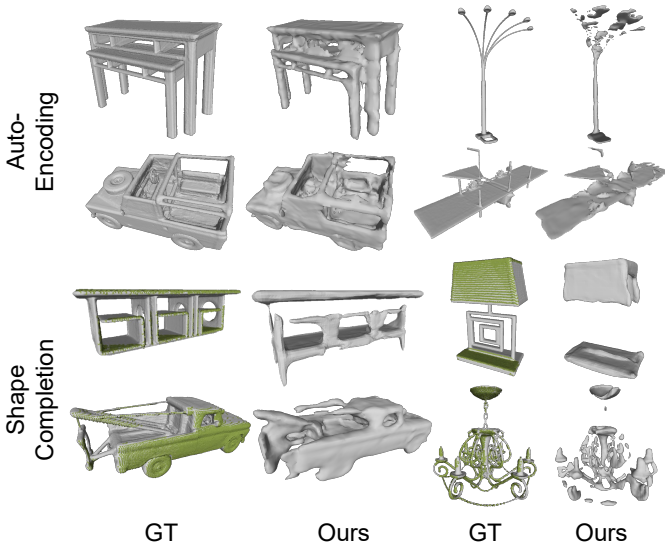


Figure 17: **Failure cases.** Row 1 and 2: auto-encoding; Row 3 and 4: shape completion from depth image.

	Ours	Ours-6	Ours-7	Ours-8	Ours	Ours-6	Ours-7	Ours-8
Chamfer	0.19	0.13	0.13	0.12	0.17	0.14	0.13	0.13
F-Score	93.0	96.5	96.7	97.5	92.8	96.3	97.1	97.3

Table 7: **Auto-encoding accuracy with more levels.** Middle columns: 3D-R^{2}N^2 test set. Right columns: unseen categories. "Ours" stands for 5 levels and "Ours-N" stands for N levels.

more levels and higher resolution latent grids. For shape completion, when an unobserved part (*e.g.*, the lamp body in row 3, column 3) is completely missing in the coarse prediction from level 0, our approach is unable to synthesize such delicate structures.

6.6. Additional Ablation for Number of Levels

In the paper, we use 5 levels as it is a good balance between accuracy and efficiency. But as previously indicated, MDIF is flexible to use other number of levels. In Figure 9, we showed progressive refinement rate-distortion for levels 1-5. Here in Table 7, we further show the auto-encoding accuracy under encoder-decoder inference with up to 8 levels.

6.7. Interpolation and Retrieval in Latent Space

Figure 18 shows linear interpolation in latent space. The latent codes for the two ends are obtained with encoder-decoder auto-encoding. Figure 19 shows results for object retrieval based on latent codes (top-2 retrievals for each query object).

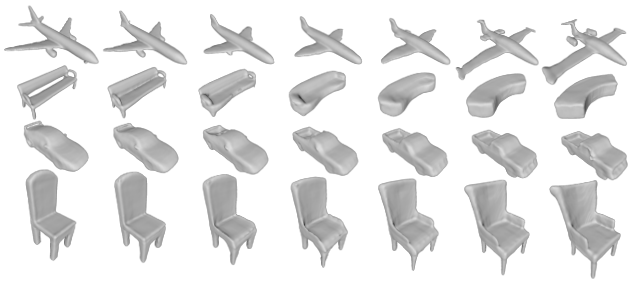


Figure 18: **Linear interpolation in latent space.**

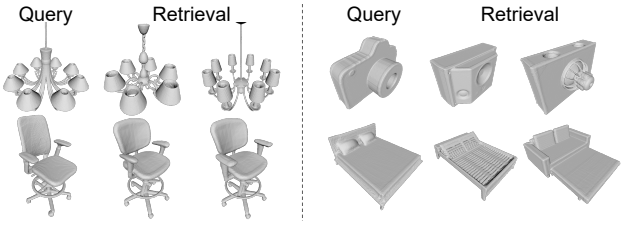


Figure 19: **Object retrieval.** Queries are from test set (left) and unseen categories (right). Retrieved objects are from training set.

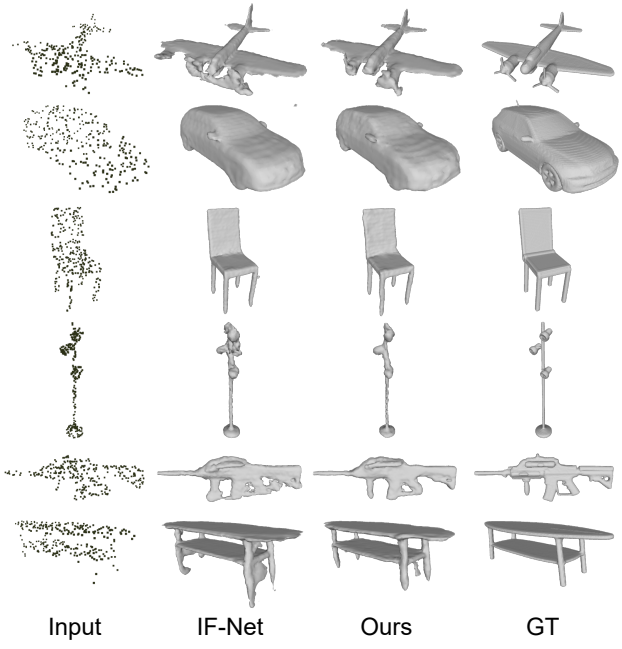


Figure 20: **Point cloud completion**. Additional qualitative results.

6.8. Additional Qualitative Results

Figure 20 and Figure 21 show additional qualitative comparisons on point cloud completion and voxel super-resolution. Compared to IF-Net, our method generally produces cleaner reconstructions with less artifacts.

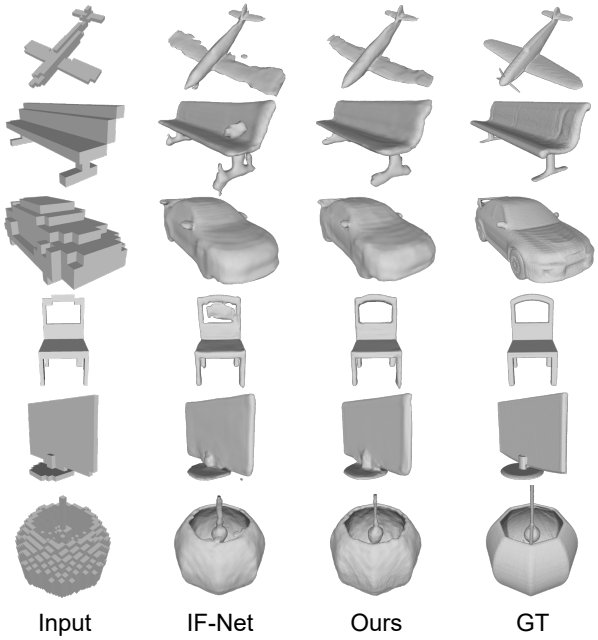


Figure 21: **Voxel super-resolution**. Additional qualitative results.

6.9. Detailed Quantitative Results

Table 8 and Table 9 show per-category quantitative results (Chamfer L2 distance and F-Score) on auto-encoding. For encoder-decoder inference, we compare MDIF with OccNet ("Occ.") [23], SIF [14], LDIF [13] and IF-Net ("IF.") [4]. For decoder-only latent optimization, we compare MDIF with OccNet ("Occ.") [23], IM-Net ("IM.") [3] and a local baseline (resembles [19, 1]). Table 10 shows per-category quantitative results (Chamfer L2 distance / F-Score) on point cloud completion and voxel super-resolution, where we compare MDIF with IF-Net [4] under encoder-decoder inference.

In these experiments, MDIF has lower Chamfer errors for most categories and higher overall F-Score.

6.10. Shape Completion User Study

First, in Table 11, we compare quantitative results of MDIF and competing methods on shape completion from depth image. In this comparison, we also include a MDIF model ("Ours") that uses encoder-decoder inference. This model has the same architecture as the MDIF model in the point cloud completion experiment, and is retrained from scratch to take voxelized depth points (depth points voxelized into a 128^3 grid) as input. In terms of metrics, we additionally use Asymmetric Chamfer to measure the reconstruction accuracy in observed regions. It is computed as one-directional Chamfer L2 distance from depth points to reconstruction.

Category	Chamfer (↓)									F-Score (↑, %)								
	Occ.	SIF	LDIF	IF.	Ours	Occ.*	IM.*	Local*	Ours*	Occ.	SIF	LDIF	IF.	Ours	Occ.*	IM.*	Local*	Ours*
airplane	0.16	0.44	0.10	0.52	**0.05**	0.25	0.13	0.044	**0.028**	87.8	71.4	96.9	94.4	**97.2**	89.8	91.7	98.5	**98.6**
bench	0.24	0.82	0.17	0.31	**0.08**	0.34	0.22	0.121	**0.052**	87.5	58.4	**94.8**	92.6	92.4	85.2	88.6	**96.0**	**96.0**
cabinet	0.41	1.10	0.33	**0.11**	0.29	0.32	0.23	0.063	**0.051**	86.0	59.3	92.0	**93.0**	91.5	83.2	89.2	**96.6**	**96.6**
car	0.61	1.08	**0.28**	0.30	0.29	0.58	0.26	0.090	**0.088**	77.5	56.6	87.2	**87.4**	86.6	69.3	82.7	**93.1**	93.0
chair	0.44	1.54	0.34	**0.10**	**0.10**	0.38	0.43	0.042	**0.035**	77.2	42.4	90.9	**94.5**	93.8	80.2	82.5	**97.7**	97.6
display	0.34	0.97	0.28	**0.07**	0.08	0.35	0.20	0.043	**0.019**	82.1	56.3	94.8	**96.1**	95.1	82.3	89.4	98.6	**98.7**
lamp	1.67	3.42	1.80	1.17	**0.90**	1.47	2.76	**0.795**	**0.795**	62.7	35.0	84.0	**89.1**	87.1	62.9	73.8	**93.5**	**93.5**
rifle	0.19	0.42	0.09	1.07	**0.05**	0.39	0.55	0.060	**0.057**	86.2	70.0	**97.3**	93.5	96.2	86.1	81.1	**96.9**	**96.9**
sofa	0.30	0.80	0.35	0.13	**0.11**	0.31	0.16	0.208	**0.037**	85.9	55.2	92.8	92.5	**93.5**	85.2	89.3	98.3	**98.4**
speaker	1.01	1.99	0.68	**0.14**	0.27	0.38	0.17	0.065	**0.044**	74.7	47.4	84.3	**90.2**	90.1	78.1	89.4	**97.3**	**97.3**
table	0.44	1.57	0.56	0.17	**0.13**	0.31	0.30	0.107	**0.046**	84.9	55.7	92.4	93.4	**93.7**	87.2	88.6	96.5	**97.6**
telephone	0.13	0.39	0.08	0.08	**0.06**	0.19	0.11	0.043	**0.010**	94.8	81.8	98.1	**98.8**	98.3	88.9	96.5	**99.6**	**99.6**
watercraft	0.41	0.78	0.20	0.90	**0.10**	0.35	0.39	0.075	**0.067**	77.3	54.2	93.2	92.7	**93.7**	80.3	84.7	**97.4**	97.2
mean	0.49	1.18	0.40	0.39	**0.19**	0.43	0.46	0.135	**0.102**	81.9	59.0	92.2	92.9	**93.0**	81.4	86.7	96.9	**97.0**

Table 8: **Per-category auto-encoding accuracy for objects in 3D-R^{2}N^2 test set of ShapeNet.** For each metric, left columns compare methods under encoder-decoder inference while right columns compare under decoder-only latent optimization. $*$: decoder-only latent optimization.

Category	Chamfer (↓)									F-Score (↑, %)								
	Occ.	SIF	LDIF	IF.	Ours	Occ.*	IM.*	Local*	Ours*	Occ.	SIF	LDIF	IF.	Ours	Occ.*	IM.*	Local*	Ours*
bed	1.30	2.24	0.68	**0.10**	0.16	0.87	0.43	0.052	**0.045**	59.3	32.0	81.4	**94.7**	90.9	67.1	77.8	96.8	**97.0**
birdhouse	1.25	1.92	0.75	0.31	**0.11**	0.72	0.49	**0.036**	**0.036**	54.2	33.8	76.2	90.4	**92.1**	61.3	74.3	97.6	**97.7**
bookshelf	0.83	1.21	0.36	0.30	**0.20**	0.99	0.60	0.103	**0.091**	66.5	43.5	86.1	**93.5**	88.3	59.0	73.0	**95.1**	94.2
camera	1.17	1.91	0.83	0.27	**0.16**	0.45	0.58	**0.047**	0.050	57.3	37.4	77.7	**95.0**	94.0	70.2	75.9	**98.6**	**98.6**
file	0.41	0.71	**0.29**	0.35	0.30	0.38	0.25	0.054	**0.041**	86.0	65.8	93.0	**95.7**	94.4	84.3	90.0	97.6	**97.7**
mailbox	0.60	1.46	0.40	1.18	**0.20**	0.51	0.74	**0.102**	**0.102**	67.8	38.1	87.6	81.4	**93.5**	80.0	85.2	**98.5**	**98.5**
piano	1.07	1.81	0.78	0.34	**0.08**	0.91	0.71	0.034	**0.030**	61.4	39.8	82.2	**96.7**	94.8	62.2	77.3	**98.3**	**98.3**
printer	0.85	1.44	0.43	**0.15**	**0.15**	0.48	0.31	**0.035**	**0.035**	66.2	40.1	84.6	**94.9**	94.3	74.9	82.3	98.2	**98.3**
stove	0.49	1.04	0.30	0.55	**0.22**	0.37	0.25	0.107	**0.040**	77.3	52.9	89.2	91.3	**93.5**	78.6	87.4	**97.7**	**97.7**
tower	0.50	1.05	0.47	0.44	**0.14**	0.53	0.30	**0.060**	0.070	70.2	45.9	85.7	90.3	**91.8**	73.9	81.7	**96.9**	**96.6**
mean	0.85	1.48	0.53	0.40	**0.17**	0.62	0.47	0.063	**0.054**	66.6	43.0	84.4	92.4	**92.8**	71.1	80.5	**97.5**	**97.5**

Table 9: **Per-category auto-encoding accuracy for objects in unseen categories of ShapeNet.** For each metric, left columns compare methods under encoder-decoder inference while right columns compare under decoder-only latent optimization. $*$: decoder-only latent optimization.

Category	Point Cloud Completion		Voxel Super-Resolution	
	IF-Net	Ours	IF-Net	Ours
airplane	2.37 / 89.7	**0.08 / 93.3**	1.51 / 78.3	**1.02 / 80.7**
bench	1.22 / 84.5	**0.18 / 86.0**	1.88 / 59.1	**1.09 / 59.5**
cabinet	1.65 / **87.1**	**0.84** / 83.8	0.65 / 60.6	**0.60 / 60.8**
car	1.96 / 79.4	**0.19 / 80.9**	0.40 / **75.8**	**0.30 / 75.8**
chair	2.02 / **81.3**	**0.33** / 80.5	1.02 / 62.6	**0.82 / 63.4**
display	1.09 / 88.5	**0.30 / 88.6**	1.04 / 62.0	**0.74 / 62.1**
lamp	2.03 / 76.3	**1.76 / 78.0**	8.14 / 58.3	**3.97 / 60.9**
rifle	2.19 / 85.3	**0.05 / 95.9**	2.09 / 78.0	**0.34 / 81.3**
sofa	0.71 / **88.2**	**0.18** / 86.8	0.68 / 56.2	**0.48 / 57.5**
speaker	1.52 / **78.4**	**0.65** / 75.9	0.73 / 56.1	**0.65 / 58.0**
table	1.70 / 84.7	**0.25 / 85.1**	2.72 / 53.5	**1.87 / 55.7**
telephone	0.98 / 95.7	**0.06 / 96.5**	0.77 / 77.9	**0.67 / 78.2**
watercraft	1.51 / 87.2	**0.14 / 88.4**	2.05 / 71.7	**0.69 / 73.6**
mean	1.61 / 85.0	**0.39 / 86.1**	1.82 / 65.4	**1.02 / 66.7**

Table 10: Per-category quantitative results (Chamfer L2 distance / F-Score) for point cloud completion and voxel super-resolution.

When comparing under encoder-decoder inference ("OccNet", "LDIF", "Ours"), MDIF is only slightly worse than LDIF in F-Score while performs the best in the other two metrics. This reveals that when using encoder-decoder inference, MDIF can produce completion results similarly close to the groundtruth as LDIF. Meanwhile, the large margin in Asymmetric Chamfer compared with OccNet and LDIF demonstrates the better capability of MDIF to preserve details in observed regions, even under encoder-decoder inference. For the MDIF model that uses decoder-only latent optimization ("Ours*"), although it has worse performance in Chamfer distance and F-Score, it can reduce the error in Asymmetric Chamfer even much further. This indicates that it performs much better on the observable parts and the source of error mostly comes from the unobserved parts. As illustrated in the paper (Figure 12), although different from

Category	Chamfer (↓)				F-Score (↑, %)				Asym. Chamfer (↓)			
	OccNet	LDIF	Ours	Ours*	OccNet	LDIF	Ours	Ours*	OccNet	LDIF	Ours	Ours*
airplane	0.47	**0.17**	0.26	0.46	70.1	89.2	**90.1**	73.2	0.246	0.054	0.022	**0.007**
bench	0.70	**0.39**	0.45	0.96	64.9	81.9	**82.5**	56.9	0.281	0.108	0.049	**0.012**
cabinet	1.13	0.77	**0.73**	1.35	70.1	**77.9**	73.8	60.4	0.109	0.052	0.070	**0.009**
car	0.99	0.51	**0.41**	1.04	61.6	72.4	**74.3**	64.2	0.138	0.054	0.043	**0.011**
chair	2.34	1.02	**0.91**	1.42	50.2	69.6	**72.5**	67.0	0.785	0.270	0.053	**0.012**
display	0.95	0.62	**0.56**	1.69	62.8	**80.0**	76.7	55.4	0.312	0.217	0.056	**0.007**
lamp	9.91	2.15	**1.26**	3.26	44.1	66.4	**70.5**	54.6	10.80	1.429	0.160	**0.110**
rifle	0.49	**0.14**	0.31	0.62	66.4	**92.3**	91.5	75.9	0.246	0.048	0.022	**0.005**
sofa	1.08	0.83	**0.70**	1.19	61.2	**71.7**	71.4	62.1	0.155	0.074	0.059	**0.007**
speaker	3.50	1.48	**1.45**	3.73	52.4	**67.3**	64.6	49.8	0.280	0.115	0.077	**0.020**
table	2.49	1.14	**0.94**	1.11	66.7	**78.0**	77.8	61.5	0.784	0.339	0.065	**0.015**
telephone	0.35	**0.19**	0.21	1.05	86.1	**92.0**	89.4	55.9	0.089	0.046	0.046	**0.002**
watercraft	1.15	0.50	**0.45**	0.69	54.5	77.5	**78.3**	67.2	0.684	0.148	0.033	**0.020**
mean	1.97	0.76	**0.67**	1.43	62.4	**78.2**	78.0	61.9	1.147	0.227	0.058	**0.018**

Table 11: **Shape completion from depth image.** Quantitative comparisons on Chamfer distance, F-Score and Asymmetric Chamfer distance. "Ours*" achieves the lowest error on the observed part, measured by the Asymmetric Chamfer distance. Its worse Chamfer and F-Score results are caused by the unobserved part. See our user study for more in-depth analysis. $*$: decoder-only latent optimization.

the groundtruth, the unobserved parts of its results still look plausible.

To prove our point, we conducted a user study to compare human subjective verdicts and F-Score. We recruited 88 participants who were at least 18 years old. All participants had no prior knowledge of this project. Each participant was given 32 pairs of examples, one from MDIF (with decoder-only latent optimization) and one from LDIF [13]. Order of the examples is fully counterbalanced and randomized. Each example was shown in two different views: one observed (input view) and one unobserved. Participants were then asked to choose which example was the more plausible reconstruction given the input. If both examples looked similarly plausible, they were allowed to choose *cannot decide*.

Examples were chosen in this way. The worst results in F-Score were filtered, since both human and F-Score tend to agree on those cases. Then examples with unmatched input views were removed. We then randomly picked 32 examples from the rest.

The results of user study are summarized in Figure 22. In contrast to F-Score, 54.2% of the participants chose in favor of MDIF results, whilst 31.9% thought LDIF results were better. In addition, 13.9% could not decide between MDIF and LDIF. Moreover, when compared with the quantitative metrics, 68.1% disagree with Chamfer L2 distance, and 51.4% disagree with F-Score. All the 32 examples and itemized results are shown in Figure 23, Figure 24, Figure 25 and Figure 26.

The conclusion of this user study aligns with previous work [33], where Chamfer distance has been argued as not suitable for evaluating completion tasks due to its sensitivity to outliers. Moreover, this study also shows that, although more robust, F-Score only tells us how different the reconstruction of the unobserved part is from the groundtruth, but not how *plausible* it is, which is what humans ultimately care about.

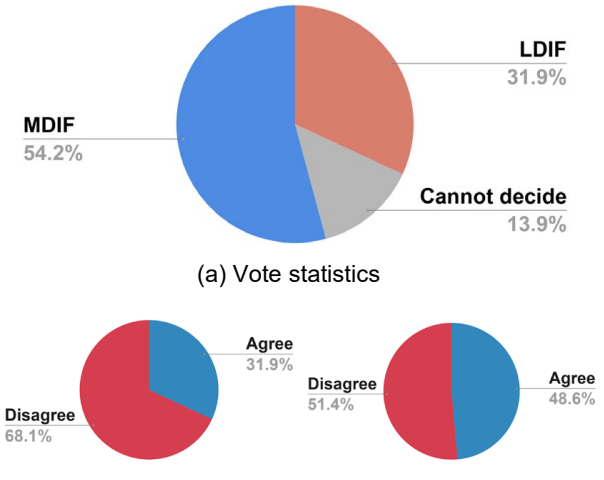


Figure 22: **Summary of user study.** Participants were asked which reconstruction was more plausible. 54.2% chose MDIF while 13.9% cannot decide between the results. Moreover, 68.1% of the votes disagree with Chamfer L2 distance, and 51.4% disagree with F-Score. Refer to Figure 23 to Figure 26 for itemized results.

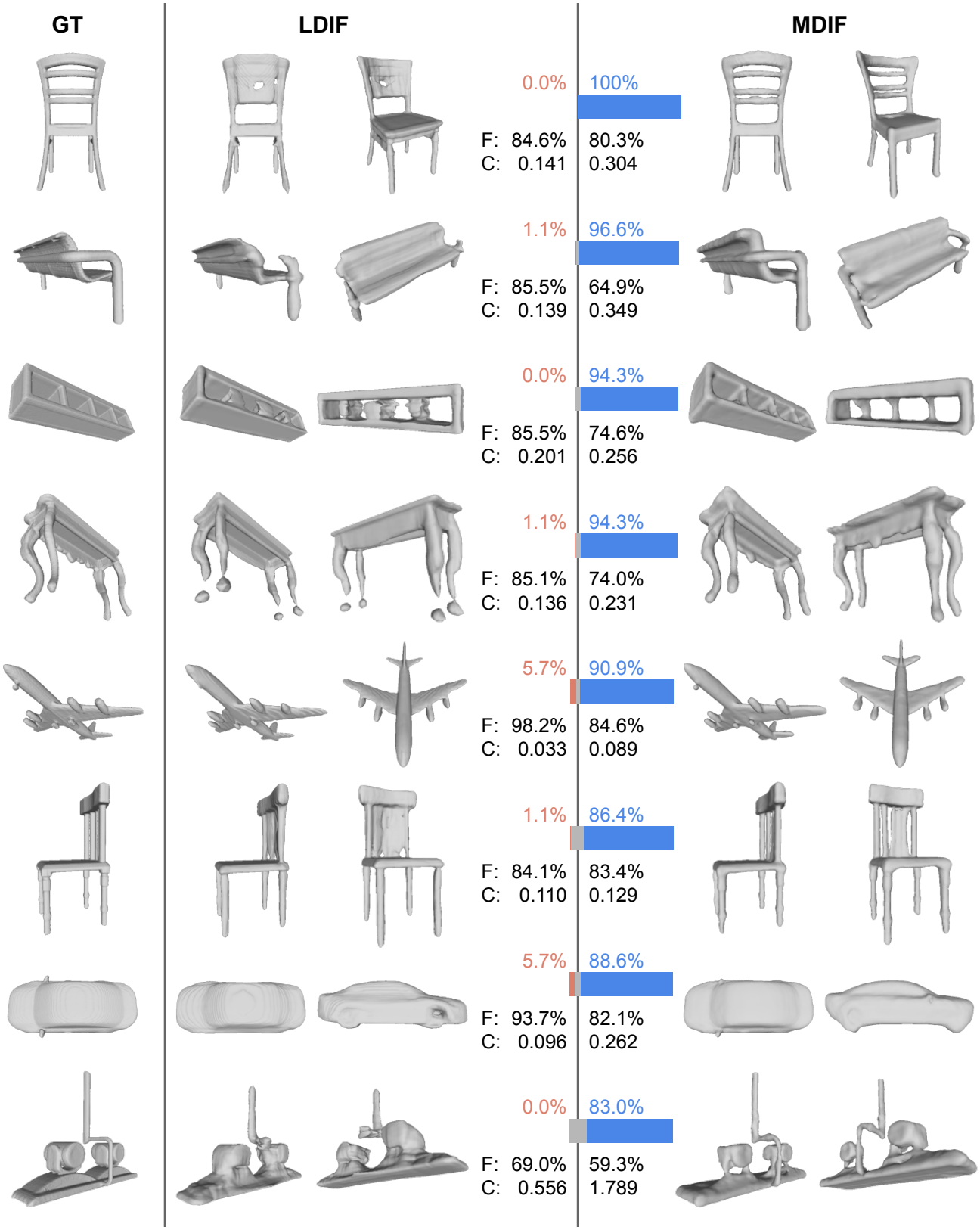


Figure 23: **Itemized user study results.** For each example, we show the groundtruth mesh under input view, and the reconstruction results under two views: one observed view same as input and one unobserved view. The bar chart shows the percentages of votes. Red: prefer LDIF; Blue: prefer MDIF; Gray: Cannot decide; F: F-Score; C: Chamfer L2 distance.

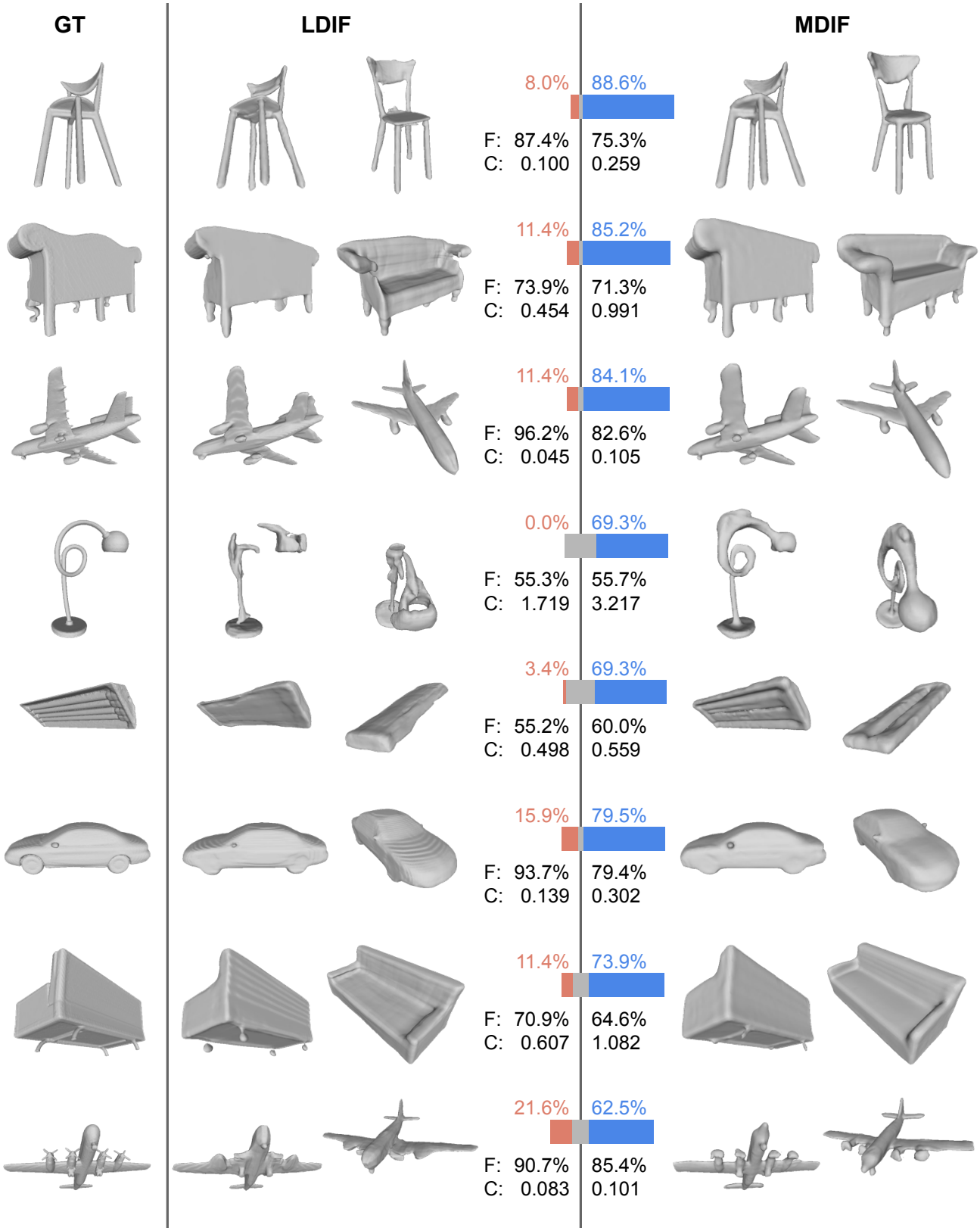


Figure 24: **Itemized user study results.** For each example, we show the groundtruth mesh under input view, and the reconstruction results under two views: one observed view same as input and one unobserved view. The bar chart shows the percentages of votes. Red: prefer LDIF; Blue: prefer MDIF; Gray: Cannot decide; F: F-Score; C: Chamfer L2 distance.

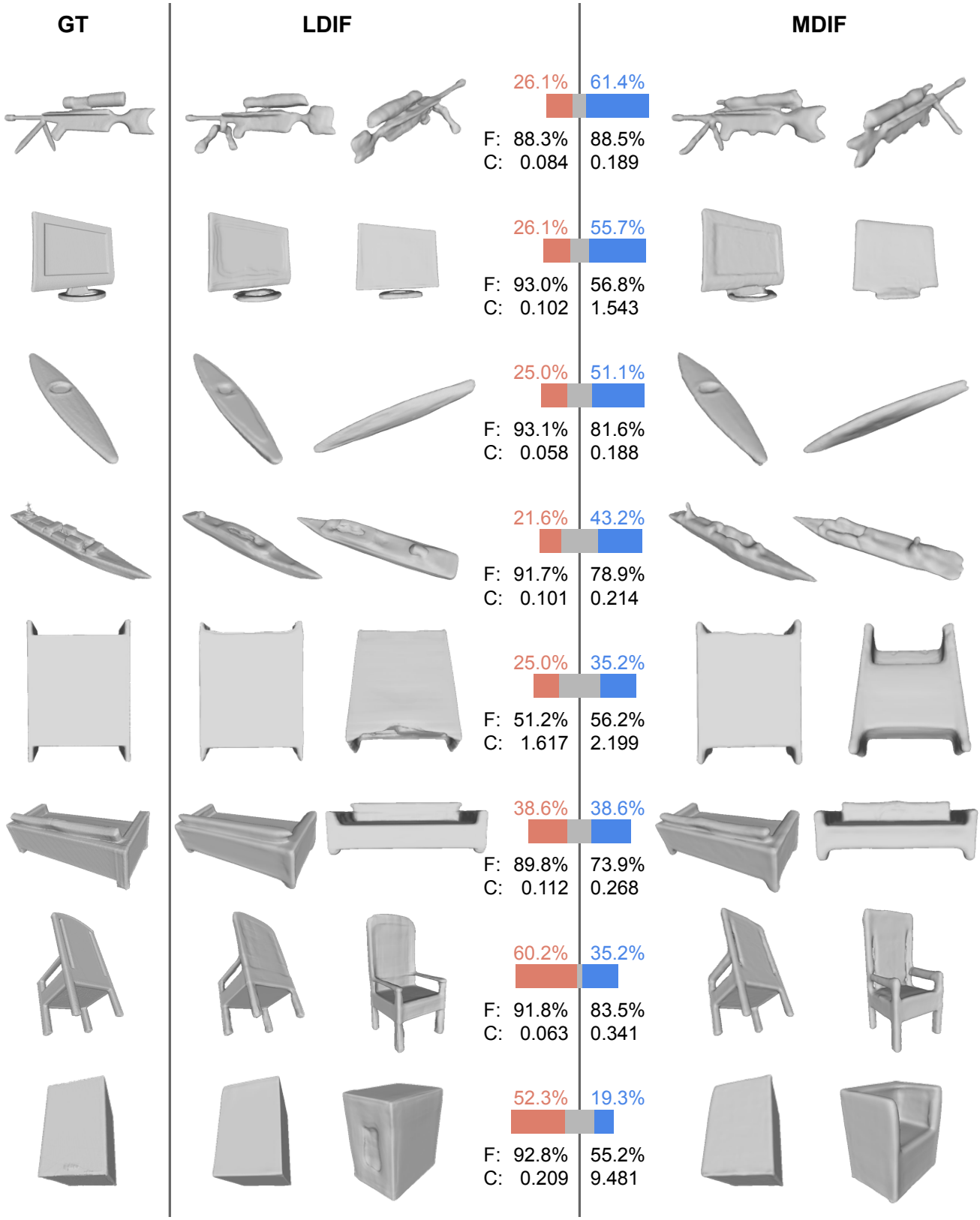


Figure 25: **Itemized user study results.** For each example, we show the groundtruth mesh under input view, and the reconstruction results under two views: one observed view same as input and one unobserved view. The bar chart shows the percentages of votes. Red: prefer LDIF; Blue: prefer MDIF; Gray: Cannot decide; F: F-Score; C: Chamfer L2 distance.

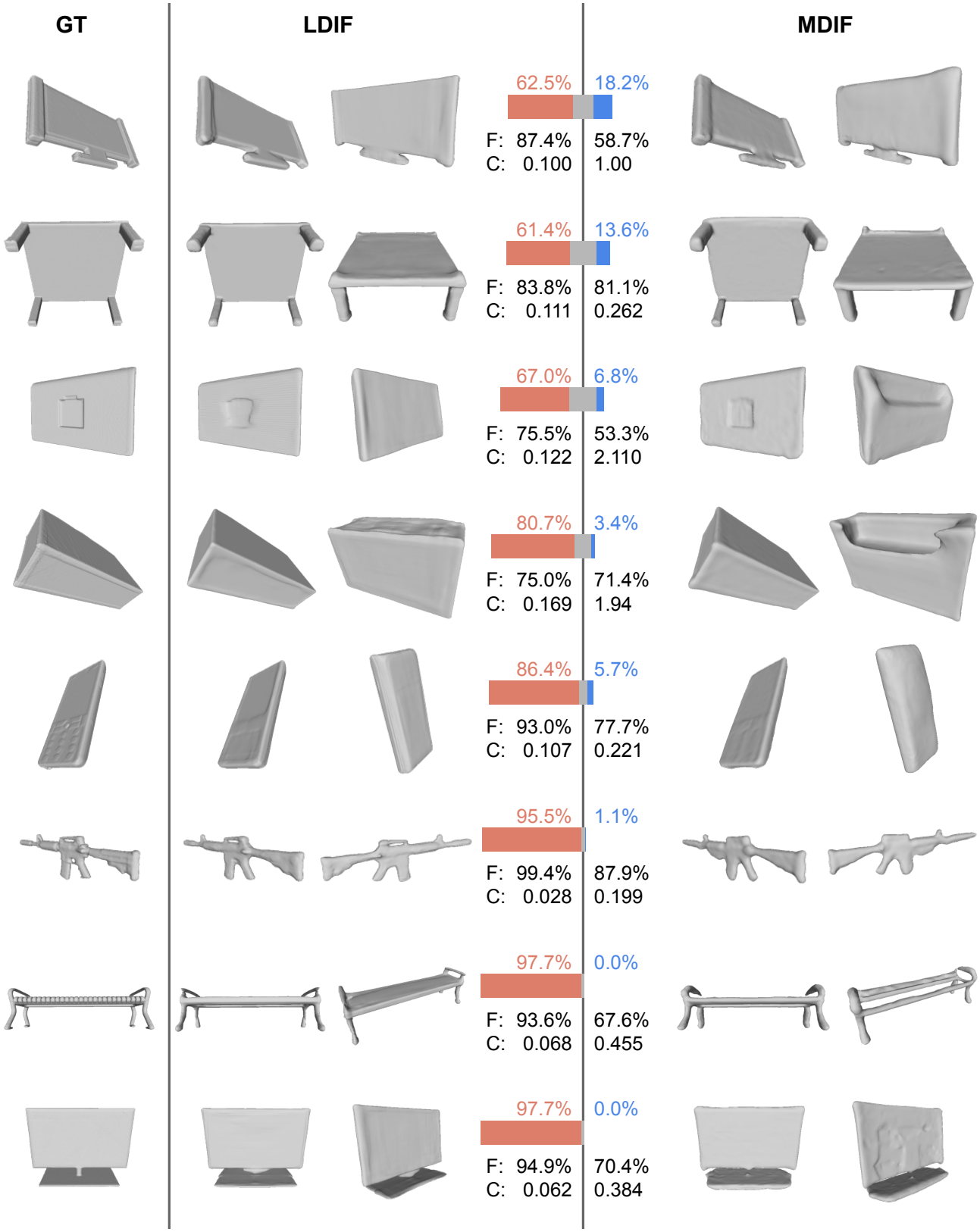


Figure 26: **Itemized user study results.** For each example, we show the groundtruth mesh under input view, and the reconstruction results under two views: one observed view same as input and one unobserved view. The bar chart shows the percentages of votes. Red: prefer LDIF; Blue: prefer MDIF; Gray: Cannot decide; F: F-Score; C: Chamfer L2 distance.

References

[1] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *Eur. Conf. Comput. Vis.*, pages 608–625. Springer, 2020.

[2] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[3] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5939–5948, 2019.

[4] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6970–6981, 2020.

[5] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Eur. Conf. Comput. Vis.*, pages 628–644. Springer, 2016.

[6] Charles K Chui. *An introduction to wavelets*. Elsevier, 2016.

[7] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996.

[8] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5868–5877, 2017.

[9] Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, , and Shahram Izadi. Motion2fusion: Real-time volumetric performance capture. *ACM TOG (SIGGRAPH Asia)*, 2017.

[10] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. Fusion4d: real-time performance capture of challenging scenes. *ACM Trans. Graph.*, 35(4):114, 2016.

[11] Ruofei Du, Ming Chuang, Wayne Chang, Hugues Hoppe, and Amitabh Varshney. Montage4D: Real-Time Seamless Fusion and Stylization of Multiview Video Textures. *Journal of Computer Graphics Techniques*, 8(1):1–34, Jan. 2019.

[12] Yueqi Duan, Haidong Zhu, He Wang, Li Yi, Ram Nevatia, and Leonidas J Guibas. Curriculum deepsdf. In *Eur. Conf. Comput. Vis.*, pages 51–67. Springer, 2020.

[13] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2020.

[14] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Int. Conf. Comput. Vis.*, pages 7154–7164, 2019.

[15] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *Eur. Conf. Comput. Vis.*, pages 484–499. Springer, 2016.

[16] Christian Häne, Sohubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(6):1348–1361, 2019.

[17] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Trans. Graph.*, 38(4):1–12, 2019.

[18] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proc. UIST*, 2011.

[19] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6001–6010, 2020.

[20] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1251–1261, 2020.

[21] Marian Kleineberg, Matthias Fey, and Frank Weichert. Adversarial generation of continuous implicit shape representations. *arXiv preprint arXiv:2002.00349*, 2020.

[22] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *arXiv preprint arXiv:2007.11571*, 2020.

[23] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4460–4470, 2019.

[24] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Deep level sets: Implicit surface representations for 3d shape inference. *arXiv preprint arXiv:1901.06802*, 2019.

[25] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Eur. Conf. Comput. Vis.*, 2020.

[26] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015.

[27] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 165–174, 2019.

[28] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. Octnetfusion: Learning depth fusion from data. In *International Conference on 3D Vision (3DV)*, pages 57–66. IEEE, 2017.

[29] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[30] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 2020.

[31] Danhang Tang, Saurabh Singh, Philip A Chou, Christian Hane, Mingsong Dou, Sean Fanello, Jonathan Taylor, Philip Davidson, Onur G Guleryuz, Yinda Zhang, Shahram Izadi, Andrea Tagliasacchi, Sofien Bouaziz, and Cem Keskin. Deep implicit volume compression. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1293–1303, 2020.

[32] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2088–2096, 2017.

[33] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3405–3414, 2019.

[34] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, Rohit Pandey, Sean Fanello, Gordon Wetzstein, Jun-Yan Zhu, Christian Theobalt, Maneesh Agrawala, Eli Shechtman, Dan B Goldman, and Michael Zollhoefer. State of the art on neural rendering. In *Eurographics*, 2020.

[35] Hao Wang, Nadav Schor, Ruizhen Hu, Haibin Huang, Daniel Cohen-Or, and Hui Huang. Global-to-local generative model for 3d shapes. *ACM Trans. Graph.*, 37(6):1–10, 2018.

[36] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Trans. Graph.*, 36(4):72:1–72:11, July 2017.

[37] Peng-Shuai Wang, Yang Liu, and Xin Tong. Deep octree-based cnns with output-guided skip connections for 3d shape and scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 266–267, 2020.

[38] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1912–1920, 2015.

[39] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems*, pages 492–502, 2019.