Secrets of 3D Implicit Object Shape Reconstruction in the Wild

Shivam Duggal[*1], Zihao Wang[†*1], Wei-Chiu Ma[1,3],
Sivabalan Manivasagam[1,2], Justin Liang[1], Shenlong Wang[1,2] and Raquel Urtasun[1,2]
[1]Uber ATG, [2]University of Toronto, [3]Massachusetts Institute of Technology

Abstract

Reconstructing high-fidelity 3D objects from sparse, partial observation is of crucial importance for various applications in computer vision, robotics, and graphics. While recent neural implicit modeling methods show promising results on synthetic or dense datasets, they perform poorly on real-world data that is sparse and noisy. This paper analyzes the root cause of such deficient performance of a popular neural implicit model. We discover that the limitations are due to highly complicated objectives, lack of regularization, and poor initialization. To overcome these issues, we introduce two simple yet effective modifications: (i) a deep encoder that provides a better and more stable initialization for latent code optimization; and (ii) a deep discriminator that serves as a prior model to boost the fidelity of the shape. We evaluate our approach on two real-wold self-driving datasets and show superior performance over state-of-the-art 3D object reconstruction methods.

1. Introduction

Consider the street view image and the partial LiDAR scan in Fig. 1. As humans, we can effortlessly identify the vehicle in the scene and have a rough grasp of its 3D geometry. We can go even a step further and answer more challenging questions like: what may the vehicle look like from a different point of view? What is the exact 3D shape of the object? This is because human visual systems have accumulated thousands hours of observations that help us develop mental models for these objects [27, 26]. While we may have never seen this particular car before, we know that cars shall be symmetric, they shall lie on the ground, and sedans shall have similar shapes and sizes. We can thus exploit these knowledge to infer the 3D structure of the instances. The goal of this work is to equip computational visual machines with similar capabilities.

Reconstructing accurate, high-fidelity 3D shape from partial observations is, however, extremely challenging.

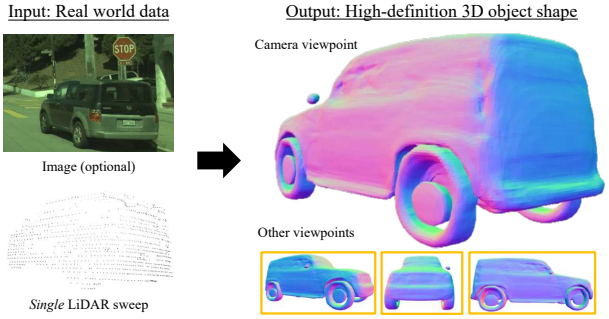


Figure 1: **3D Object Reconstruction in the Wild.** Our model takes as input a single LiDAR sweep and optionally a RGB image, and outputs an accurate, high-fidelity 3D shape.

The problem is inherently ill-posed and one has to deal with the potential noises, occlusions, and environment variability presented in the data. To tackle these issues, previous approaches usually take one of the following two routes: (i) learn a direct mapping from observations to the output 3D space; or (ii) model the 3D object reconstruction problem as a structured optimization task and incorporate human knowledge into the model. Specifically, former methods capitalize on machine learning algorithms to directly learn the statistical priors from data. While they are more robust to the noises presented in the observation, the predicted outputs are not guaranteed to match with the input observations and are often not structured. Additionally, many feed-forward methods are learned with ground-truth supervision from synthetic data, which is often unavailable for real-world data. This leads to poor generalization performance on unseen partial inputs. In contrast, optimization based approaches can produce coherent 3D shapes through imposing carefully designed objectives. In practice, however, the design process is cumbersome and almost no hand-crafted priors can include all possible phenomena. Despite hand-crafted regularization, optimization-based methods struggle with noise and create unusual artifacts when trying to be consistent with sensor observations (see Fig. 2). Both approaches have yet to generate the quality of shapes observed on synthetic data to the real-world domain.

[†]Work done as part of internship.

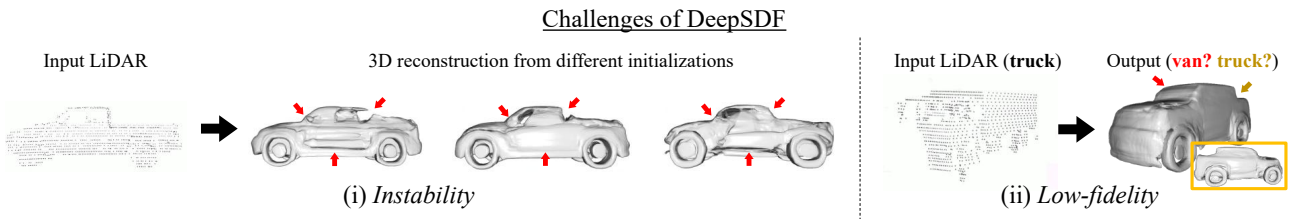


Figure 2: **Challenges of DeepSDF.** (i) *Instability*: while the input observation remains intact, different initializations may lead to distinct 3D reconstructions. See the red arrows. (ii) *Low-fidelity*: the model may easily overfit to the noise in the data, and fail to capture the global structure of the input observation. Here, the output shape is a hybrid of a van and a truck.

The goal of this work is to bring the best of both worlds. We build our model based upon the popular DeepSDF [32]. DeepSDF can be viewed as an optimization based method that combines the shape priors encoded in neural networks with structured optimization. Instead of explicitly representing the surface, DeepSDF learns an implicit function in the continuous space to produce high-quality 3D shapes. While it has achieved state-of-the-art performance on *synthetic* datasets such as ShapeNet [6] or datasets with *dense* observations such as ScanNet [9], the method suffers from 3D object reconstruction in the wild, in particular when the observation is limited and sparse. Fig. 2 shows an example where the model becomes *instable*, and the reconstruction results have *low-fidelity*.

To this end, we analyze the root cause of *instability* and *low-fidelity* of neural implicit models [32, 40, 45], and present two simple modifications. Specifically, we discover that the instability of prior art is due to the highly complicated and non-convex nature of the objective function induced by deep decoder. Furthermore, the low-fidelity is due to the lack of structured regularization in the energy functions. To overcome these issues, we introduce: (i) a deep encoder which maps the input sensor observations to an initial latent code; and (ii) a deep discriminator that regularizes the latent code to be in the object category's domain. By initializing from a better latent code and regularizing with a powerful prior model, our proposed model significantly boosts the fidelity and stability. Please refer to Fig. 3 for a detailed comparison between DeepSDF and our method.

We evaluated our approach for 3D vehicle reconstruction in the wild on two challenging self-driving datasets. We compare against state-of-the-art models on three tasks: LiDAR-based shape completion, image-based shape reconstruction and image + LiDAR shape completion. Our results show superior performance in terms of both reconstruction accuracy and visual quality.

2. Related Work

Feed Forward Shape Completion: Deep feed-forward learning approaches encode images or depth representations into latent representations, which are subsequently decoded into complete meshes or point clouds. Early work used voxel-grid based representations [11, 23] which had lossy resolution and memory constraints. Other works instead leveraged point clouds directly to create latent codes [34, 35, 20] for point cloud completion or mesh construction, using a variety of decoder architectures [60, 61, 29, 48, 55], enhanced intermediate representations [58, 18], and GAN-based loss functions [39]. While being successful on synthetic data [6] or dense 3D scans [9], works that have applied their methods [61, 58, 20] to real world noisy datasets such as KITTI [15] have had limited success. Other works [17, 10] directly perform mesh prediction from images and 3D data. Recent work [45, 30, 37, 59, 38, 49] represent implicit functions as neural networks for shape completion and have shown promising results on synthetic datasets [6, 1] or dense scans [5], but few have shown large-scale results on real data [20]. [46, 31] attempted to adapt feed-forward approaches trained on synthetic data to real data by training an encoder that consumes partial point clouds and produces latent codes that match with the observation. However, to our knowledge, most previous feed-forward works applied to real sparse data generate shapes that are often amorphous, overly smoothed, or restricted to small vehicles.

Optimization-based Shape Completion: Another line of work frame shape completion as an optimization problem, where the objective is to ensure the predicted shape is consistent with real sensor observations. Such works require strong shape priors, as the partial observations in real data are quite sparse, noisy, and have large holes. [12, 13, 53, 28] represent the shape prior as a PCA embedding of the volumetric signed distance fields, and optimize the shape and pose given sensor data. While showing promising performance on pose estimation and object tracking, the recovered shapes are coarse due to the low dimensional linear embedding, and most work demonstrate their approach only on sedans and smaller vehicles. For mesh-based approaches, [3] performs joint optimization on local meshlet priors to extract 3D meshes, and [24] uses an input point cloud as a prior to deform a convex hull mesh, but both require dense inputs. Recent work encode shape priors via

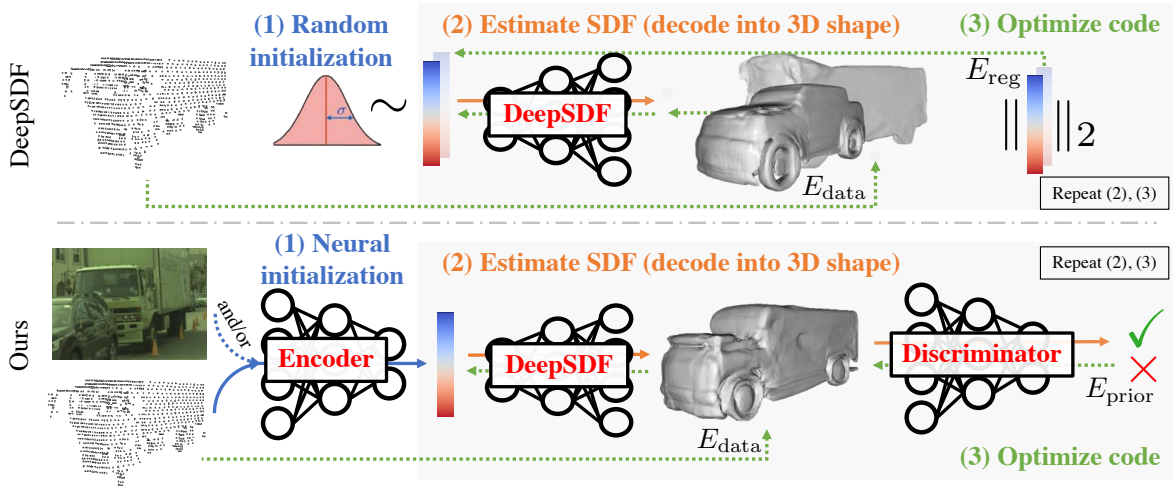


Figure 3: **Model Comparison:** (Top) DeepSDF randomly samples an initial latent code. Due to complicated objective, different initializations result in diverse results. Furthermore, due to the weak regularization, the generated shape may overfit to noise and fail to capture the global structure. (Bottom) Our model exploits a deep encoder to predict an initialization code from raw sensory data and a deep discriminator as a regularization during optimization, improving fidelity and robustness. Red arrows denote forward pass, Green arrows denote backward pass.

neural implicit representations, and perform latent code optimization to input observations [32], with some using differentiable rendering in optimization [40, 25]. These works have demonstrated success in shape completion quality, but focus on synthetic data. [63] have explored this approach for autolabeling, but do not focus on the fidelity.

Latent Space Optimization: Optimizing network weights or latent codes at test time has been quite popular for several tasks [4, 8, 64, 32]. CodeSLAM [4] optimizes geometric and motion codes during inference time for VisualSLAM, Chen *et al*. [8] optimizes geometric and appearance codes at inference time for pose estimation. Recently, latent code optimization has been explored in the field of GAN inversion [64, 41, 21] in order to determine the latent-code used to generate and manipulate the realistic images. In this work, we take inspiration from the theory of GAN inversion and apply it on the task of high-fidelity shape reconstruction. Another work related to ours is [22], which performs latent code optimization for point cloud completion and uses a GAN to ensure the latent code optimized is in the generator's manifold. However, their approach was demonstrated mostly on synthetic data or data captured in a controlled setting. In contrast, we demonstrate our approach on large-scale real-world datasets with improved performance over state-of-the-art.

3. Background

Let $\mathbf{o} \in \mathcal{O}$ be the observation obtained from the raw sensory data. The goal of 3D object shape reconstruction is to recover the underlying 3D geometry $\mathbf{s} \in \mathcal{S}$ of an object from $\mathbf{o}$. While there are various ways to represent the 3D shape, such as exploiting 3D point clouds [14, 34, 54], triangulated meshes [51, 19], 3D voxels [57, 56, 11, 44], or octo-trees [52, 36], in this work, we parameterize the 3D shape with implicit functions $\mathbf{s} = \{(\mathbf{x}, s) | (\mathbf{x}, s) \in \mathbb{R}^3 \times \mathbb{R}\}$. The value s indicates whether a point is inside (negative) or outside (positive) of the watertight surface, and how far it is to the closest surface. If $s = 0$, the point is on the surface. Such a representation is not only powerful and flexible, but also easy to learn [43, 42]. Our goal is to predict the signed distance value $s \in \mathbb{R}$ for an arbitrary 3D point $\mathbf{x} \in \mathbb{R}^3$, conditioned on the input $\mathbf{o}$. As for the input, we exploit either the sparse, noisy LiDAR point cloud $\mathbf{P} \in \mathbb{R}^{N \times 3}$, the captured RGB image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, or both $\mathbf{o} = \{\mathbf{P}, \mathbf{I}\}$.

3.1. Feed-forward Networks

One straightforward way is to directly learn a mapping from the observation $\mathbf{o}$ to its signed distance function s: $s = f(\mathbf{x}, \mathbf{o})$. These approaches [30, 59, 37, 38, 49] often employ an encoder-decoder architecture, where the goal of the encoder is to extract discriminative visual cues $\mathbf{z}$ from the input observation, and the role of the decoder is to transform the latent feature to the final 3D output:

$$s = f(\mathbf{x}, \mathbf{o}) = f_{\psi}^{\text{dec}}(\mathbf{x}, f_{\phi}^{\text{enc}}(\mathbf{o})). \quad (1)$$

Here, ψ and ϕ are the learnable parameters. Since the encoder f^{enc} mainly abstracts global structural information into the latent code $\mathbf{z} = f_{\phi}^{\text{enc}}(\mathbf{o})$, it makes the models more robust to noises presented in input. While these approaches

have demonstrated impressive performance on various challenging tasks such as single image 3D reconstruction [30], human digitization [37], etc, they have difficulty handling out-of-distribution examples, which can cause unreasonable outputs. Furthermore, these methods cannot ensure the estimated 3D geometry is consistent with the input observation. For example, the consistency between the estimated 3D shape and the sparse LiDAR point cloud.

3.2. Neural Optimization

Another line of work focuses on combining the representation power of neural networks with structured optimization. They allow us to incorporate prior knowledge into the model and ensure better consistency to the input. One pioneering effort is DeepSDF [32]. DeepSDF is an *auto-decoder* based model which consists of only $f_\psi^{\text{dec}}(\mathbf{x}, \mathbf{z})$. During training, the model learns to *both* decode the spatial point $\mathbf{x}$ into SDF value s and assign a latent code $\mathbf{z}$ to each shape. As a result, the latent code learns to be a compact yet expressive representation of 3D shapes. At inference, the network weights ψ are fixed. The model optimizes the latent code to minimize the pre-defined energy function:

$$\mathbf{z}^* = \underset{\mathbf{z}}{\operatorname{argmin}}\, E_{\text{data}}(\mathbf{o}, f_\psi^{\text{dec}}(\mathbf{X}, \mathbf{z})) + \lambda E_{\text{reg}}(\mathbf{z}). \quad (2)$$

where $f_\psi^{\text{dec}}(\mathbf{X}, \mathbf{z}) = \text{vec}[..., f_\psi^{\text{dec}}(\mathbf{x}_i, \mathbf{z}), ...]$. The data term E_{data} ensures the consistency between the estimated 3D shape and the observation, while the regularization term E_{reg} constrains the latent code. The final 3D shape can be obtained by querying the SDF value via $f_\psi^{\text{dec}}(\cdot, \mathbf{z}^*)$.

In practice, for the shape completion [32], the input to the model is a partial point cloud $\mathbf{P}$. The data term is implemented as the clamped $\mathcal{L}_1$ distance between the estimated $f_\psi^{\text{dec}}(\mathbf{P}, \mathbf{z})$ and $\mathbf{0}$ (as point clouds lies on the object surface). The prior term is a $\mathcal{L}_2$ regularization. The latent code $\mathbf{z}$ is randomly initialized from a normal distribution during optimization. We refer the readers to [32] for more details.

Limitations: Such approaches have resulted in accurate and high-fidelity shapes on synthetic datasets. However, the model experiences a significant performance degradation when applied onto real-world data captured in the wild. Specifically, we observe two major caveats of DeepSDF: (1) *instability*: different initializations may result in different shapes at convergence; and (2) *low-fidelity*: the optimal code is not guaranteed to generate a "natural" vehicle that captures the global structure the observation presents. Fig. 2 (i) shows an example where we apply DeepSDF to reconstruct a pick-up truck from a single LiDAR sweep. We run inference from different random initializations. The resulting shapes differ among different runs and do not reflect the correct object structure.

Fig. 2 (ii) shows the low-fidelity and overly smooth shape that DeepSDF generates, as it cannot constrain the latent code to produce natural shapes.

Analysis: The first drawback arise from the fact that the deep decoder is a complicated non-linear function. When plugged into the optimization procedure (see Eq. 2), the landscape of the objective function becomes highly non-convex. A minor perturbation in the initialization may leads to completely different local minimal (final output). The second defect is due to the lack of structural regularization in the objective. Specifically, the data term $E_{\text{data}}(\mathbf{o}, f_\psi^{\text{dec}}(\mathbf{X}, \mathbf{z}))$ in Eq. 2 is typically decomposed into the sum of independent terms per each individual point $\sum_{\mathbf{x}\in\mathbf{X}} E_{\text{data}}(\mathbf{o}, f_\psi^{\text{dec}}(\mathbf{x}, \mathbf{z}))$. There is little constraint on the global SDF field — each data point operates individually. The model can thus easily overfit to the noise it has never seen during training and the latent code may drop out of the manifold during optimization, leading to artifacts. This issue is particularly severe when observation is partial or sparse, which is the case for many real-world scenarios. Additionally, $E_{\text{reg}}(\mathbf{z})$ is simply a L2-normalization, with an underlying assumption that the latent code follows Gaussian distribution. Such a simple assumption may not correctly reflect the real-world latent code distribution.

4. Method

In this section, we present a simple approach for 3D object shape reconstruction in the wild. We build our model based on the observation that feed-forward networks and neural optimization are complementary — one is good at extracting robust, discriminative cues that can be transferred, while the other excels at inducing human priors and enforcing consistency with the observation. Towards this goal, we first propose two simple modifications to overcome the limitations of neural optimization approaches discussed in Sec. 3.2. Then we show how to perform effective inference using these modifications. Finally, we showcase our curriculum training strategy that help enable our approach to perform properly on real world data.

4.1. Neural Initialization & Regularization

To prevent DeepSDF from failing catastrophically when applied to real-world data, we introduce two additional modules into the current deep optimization paradigm: an encoder f^{enc} and a discriminator D.

Encoder As Initialization: Recall the instability of the deep optimization approach. The goal of the encoder is to provide a robust initialization code for DeepSDF from real-world data (*e.g.* noisy, sparse point clouds), *i.e.*, $\mathbf{z}^{(0)} = f^{\text{enc}}(\mathbf{o})$. The estimated code shall not only lie on the shape

Method	ACD (mm) ↓	Recall (%) ↑
ONet [30]	22.76	49.56
GRNet [58]	12.70	77.59
SAMP [13]	176.42	65.58
DIST [40]	19.55	71.54
DIST++ [40]	17.29	72.50
DeepSDF [32]	8.34	84.71
Ours	**5.93**	**88.18**

Table 1: **LiDAR Completion Results on NorthAmerica.** Thanks to the strong shape prior, our approach can be trained and optimized on real-world dataset effectively. We reduce the ACD error by 50% compared to the best deep feed-forward network GRNet and more than 25% compared to the current best deep optimization method DeepSDF.

manifold of DeepSDF, but also be adjacent to the optimal code $\|\mathbf{z}^{(0)} - \mathbf{z}^*\| < \epsilon$ so that even without further optimization the shape reconstruction still has high quality. We adopt Point Completion Network (PCN) [61]'s encoder as our encoder architecture. The PCN encoder computes both local point-based features and global features and combines them to get the final latent representation. This representation has shown to be robust to the noise in real-world data.

Discriminator As Prior: Based on the observation that neural optimization based approaches lack a strong prior model, we introduce a discriminator to induce a learned structured prior over the SDF field generated by DeepSDF *as a whole*. Specifically, we introduce a discriminator function D_θ parameterized by θ. Similar to deep image prior [50], such a function outputs a quantity evaluating the likelihood of a input shape for being a natural shape. To capture the overall geometry of the SDF field, we randomly sample N points from the 3D space, query their SDF value, and pass all of them into the discriminator, which outputs a single scalar representing the "naturalness" of the shape. By encouraging the produced results to be indistinguishable from the target one (*e.g.* clean synthetic data), we can prevent DeepSDF from overfitting to the noisy point clouds. We employ PointNet [33] as our discriminator architecture.

4.2. Inference

With the help of the two networks, at inference time, we reason the optimal latent code of the shape by minimizing the following objective function:

$$\mathbf{z}^* = \underset{\mathbf{z}}{\operatorname{argmin}}\, E_{\text{data}}(\mathbf{o}, f_\psi^{\text{dec}}(\mathbf{X}, \mathbf{z})) + \lambda_{\text{reg}} E_{\text{reg}}(\mathbf{z}) + \lambda_{\text{prior}} E_{\text{prior}}(f_\psi^{\text{dec}}(\mathbf{X}, \mathbf{z})). \tag{3}$$

Our data term E_{data} has the same format as DeepSDF:

$$E_{\text{data}}(\mathbf{o}, f_\psi^{\text{dec}}(\mathbf{X}, \mathbf{z})) = \sum_i^N \rho(s_i, f_\psi^{\text{dec}}(\mathbf{x}_i, \mathbf{z}))$$

where s_i is the actual signed distance value at point $\mathbf{x}_i$, computed from the observation $\mathbf{o}$. ρ is a robust clamped ℓ_1-norm. The prior term $E_{\text{prior}}(f_\psi^{\text{dec}}(\mathbf{X}, \mathbf{z}))$ encodes the belief of the discriminator that the generated shape is realistic:

$$E_{\text{prior}}(f_\psi^{\text{dec}}(\mathbf{X}, \mathbf{z})) = -\log D_\theta(\mathbf{X}, f_\psi^{\text{dec}}(\mathbf{X}, \mathbf{z}))$$

The regularization term equals to ℓ_2-norm of the latent code (*i.e.*, $E_{\text{reg}}(\mathbf{z}) = \|\mathbf{z}\|_2^2$), in order to avoid under-fitting

We minimize such energy using first-order gradient-based optimizer, where the initialization of the latent code $\mathbf{z}^{(0)} = f^{\text{enc}}(\mathbf{o})$ is obtained from the encoder.

4.3. Learning

Learning the encoder, decoder and discriminator jointly is challenging because of the inter-dependence between the modules. Inspired from the previous works [16, 56], we fragment our training procedure into three stages. In Stage 1, we train the decoder on clean and dense synthetic data, inducing a strong shape prior. Then, to ensure our approach can handle sparse inputs, in Stage 2 we train the encoder and discriminator components on sparse synthetic data. Training on sparse synthetic data provides strong supervision since we know the ground-truth shape. Finally, in Stage 3 we adapt our encoder network to real world input sparse data. We now discuss each training stage in more detail.

Stage 1: We train the decoder over synthetic training data, in the same manner as DeepSDF. Specifically, we jointly optimize the decoder weights and latent code to reconstruct the ground truth (GT) signed distances:

$$\min_{\{\mathbf{z}_i\},\psi} \mathcal{L}^{\text{dec}} = \min_{\{\mathbf{z}_i\},\psi} \sum_i^M \sum_j^N \rho(s_{i,j}, f_\psi^{\text{dec}}(\mathbf{x}_{i,j}, \mathbf{z}_i))$$

where M is the total number of training shapes, $s_{i,j}$ is GT signed distance function for a training point sample $\mathbf{x}_{i,j}$, and N is the number of training point sample per shape. Once f_ψ^{dec} is trained, we keep it fixed, preserving its ability to generate "natural" shapes. The optimized latent codes and the predicted signed distance fields are used as pseudo-ground-truth (pseudo-GT) in the next stage.

Stage 2: Next, we train the encoder and discriminator networks jointly on the synthetic dataset, keeping the decoder of the previous stage fixed. The input to the encoder is a synthetic partial point cloud, which is sampled from a

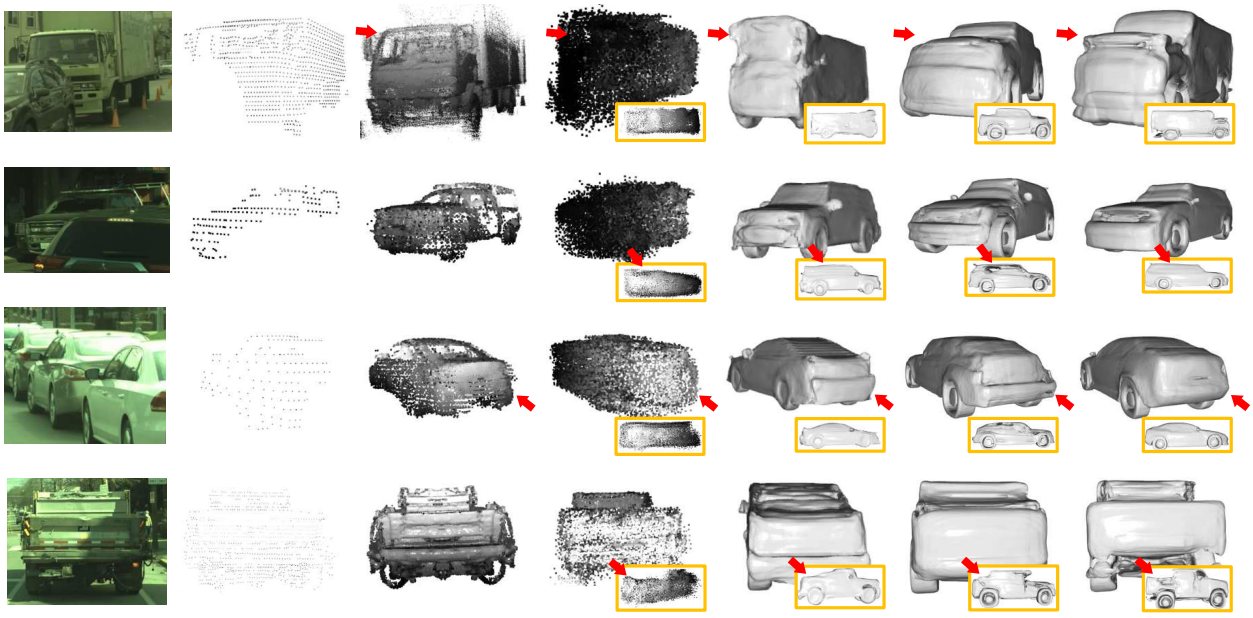


Figure 4: **LiDAR Completion Visualization on NorthAmerica:** In comparison with the baselines, our approach: (Row 1) captures the global structure, (Row 2) performs well even in occluded scenarious, (Row 3) registers well with the input and the GT point cloud and (Row 4) maintains finer details (see the cavity at the back of the car). Check supp. for more visual comparison.

ground-truth synthetic mesh following the real-world point cloud empirical distributions. This helps to reduce the synthetic-to-real domain gap in the next stage. Given this synthetic input partial point cloud, the encoder is trained to predict a latent code. This latent code is used by the decoder to predict a signed distance value for each query point. The discriminator then judges the naturalness of the decoder predicted signed distance field as a whole by comparing it with the corresponding pseudo-GT signed distance field (from Stage-1). The overall loss function for training is:

$$\mathcal{L} = \mathcal{L}^{\text{dec}} + \mathcal{L}^{\text{dis}} + \mathcal{L}^{\text{z}}$$

where $\mathcal{L}^{\text{dec}}$ measures the clamped $\mathcal{L}_1$ distance between the estimated signed distance value and the GT signed distance value for each point (same as $\mathcal{L}^{\text{dec}}$ loss in Stage 1). $\mathcal{L}^{\text{dis}}$ is the GAN loss between the predicted SDF field and the target SDF field. $\mathcal{L}^{\text{z}}$ measures the $\mathcal{L}_2$ distance between the encoder predicted code and the target code. The target distributions for $\mathcal{L}^{\text{dis}}$ and $\mathcal{L}^{\text{z}}$ come from the corresponding predicted distributions (defined as pseudo-GT) of the previous stage. [1] Since the encoder and discriminator are trained on sparse synthetic data, their predictions on real data can act as regularization on the shape prediction in the final training stage.

Stage 3: Finally, we move on to real-world point clouds. Since we trained our encoder on the simulated synthetic dataset, we can simply fine-tune the encoder on the real-world point cloud distribution. We keep the decoder and the discriminator fixed, leveraging them as shape priors. The overall loss function is similar to the one defined in Section 4.2, except we replace the latent regularization term ($E_{\text{reg}}(\mathbf{z})$ in Eq. 3) with a supervised $\mathcal{L}^{\text{z}}$ loss in this stage. Since there is no ground truth latent code for $\mathcal{L}^{\text{z}}$ loss, we treat the output of the Stage 2 encoder (trained on synthetic sparse data) as the pseudo-GT code.

4.4. Multi-modal Extensions

Image: So far we have been focusing on encoding LiDAR point clouds. The proposed pipeline, however, can be applied to real world images as well. We simply need to train a new encoder and discriminator for images. Since the synthetic dataset we used does not have realistic texture maps for the corresponding 3D CAD models, we directly train the image encoder and discriminator on the real-world dataset. The objective function is exactly similar to the one used in Stage 2 of synthetic training. The GT signed distance values for $\mathcal{L}^{\text{dec}}$ loss comes from real-world aggregated LiDAR points. The output of the Stage 2 encoder (trained on syn-

[1] We observed that utilizing generated SDF field of Stage 1 instead of GT SDF field for GAN discriminator loss results in better training and hence better results.

Method	ACD (mm) ↓	Recall (%) ↑
ONet [30]	18.10	59.10
GRNet [58]	13.66	78.21
DeepSDF [32]	14.81	79.76
Ours	**8.60**	**82.97**

Table 2: **LiDAR Completion Results on KITTI:** DeepSDF (optimzation based) performs worse than GRNet (feed-forward based) on ACD metric because of noise/ outliers in KITTI point clouds. Our approach combines the strength of both feed-forward and optimization approaches and outperforms all the baselines.

thetic sparse data) serves as the pseudo-GT code for $\mathcal{L}^{\text{z}}$ loss, while the output SDF field of the Stage 2 decoder serves as the real sample for the GAN discriminator loss. In practice, we initialize the image encoder with ImageNet pre-trained weights.

Image+LiDAR: Suppose now we these networks trained and the two encoders $f^{\text{enc}}_{\text{img}}$ and $f^{\text{enc}}_{\text{LiDAR}}$ predict their initialization code $\mathbf{z}^{\text{init}}_{f_{\text{img}}}$ and $\mathbf{z}^{\text{init}}_{f_{\text{LiDAR}}}$ respectively. How shall we combine the two codes to conduct inference?

Inspired by the latest effort on GAN inversion [21] and photometric stereo [7], we propose to fuse the latent code at the feature level. We first divide DeepSDF into two sub-networks $f^{\text{dec}}_{l,1}$ and $f^{\text{dec}}_{l,2}$, where l indicates the layer that we split, and $f^{\text{dec}} = f^{\text{dec}}_{l,2} \circ f^{\text{dec}}_{l,1}$. Then we pass both latent codes into the first sub-network $f^{\text{dec}}_{l,1}$. Finally the estimated features are aggregated and passed into the second sub-network $f^{\text{dec}}_{l,2}$. More formally, the inference procedure becomes:

$$\begin{aligned}\mathbf{z}^* = \underset{\mathbf{z}_{\text{img}},\mathbf{z}_{\text{LiDAR}}}{\text{argmin}}\ & E_{\text{data}}(\mathbf{o}, f^{\text{dec}}_{l,2}(\mathbf{X}, \mathbf{z}_l)) + \lambda_{\text{reg}} E_{\text{reg}}(\mathbf{z}_{\text{img}}) \\ & + \lambda_{\text{reg}} E_{\text{reg}}(\mathbf{z}_{\text{LiDAR}}) + \lambda_{\text{prior}} E_{\text{prior}}(f^{\text{dec}}_{l,2}(\mathbf{X}, \mathbf{z}_l)),\end{aligned} \tag{4}$$

where $\mathbf{z}_l = g(f^{\text{dec}}_{l,1}(\mathbf{x}, \mathbf{z}_{\text{img}}),\ f^{\text{dec}}_{l,1}(\mathbf{x}, \mathbf{z}_{\text{LiDAR}}))$ is the aggregated feature and g is the aggregation function. Following [7], we adopt max-pooling to fuse the features. The advantages are three fold: (i) there are no extra parameters; (ii) the aggregation function can be extended to take multiple features as input without modification; and (iii) max-pooling over the features allows each latent code to focus on the part it is confident in and distribute the burden accordingly.

5. Experiments

In this section, we first describe our experimental setup. We then compare our proposed approach against a comprehensive set of 3D object reconstruction methods on various tasks, including LiDAR-based 3D shape completion, image-based 3D object reconstruction, and image-guided 3D shape completion.

Method	ACD (mm) ↓	Recall (%) ↑
DIST [40]	62.97	48.82
Ours	**8.89**	**84.32**

Table 3: **Monocular Image Reconstruction Results:** Thanks to the strong shape prior obtained from synthetic dataset, our approach can be trained on real-world point clouds and it significantly outperforms DIST on both metrics. Visual comaprisons are shown in supp.

5.1. Datasets

ShapeNet: We exploit 2364 watertight cars from ShapeNet [6] as our synthetic dataset. We follow DeepSDF [32] to generate the ground truth SDF samples for the first stage of training. For stage 2, we simulate the image and the sparse point clouds for each object from five different viewpoints, resulting in a total dataset of 11820 image-point cloud pairs. Camera images and sparse point clouds are rendered through Blender cycles engine, where camera viewpoints are sampled from an empirical distribution collected from our real-world self-driving data.

NorthAmerica: We further build a novel, large-scale 3D vehicle reconstruction dataset using a self-driving platform that collects data over multiple metropolitan cities in North America and under various weather conditions and time of the day. Our data consists of a sparse point cloud from a 64-line spinning 10Hz LiDAR and RGB images captured by a wide-angle perspective camera, with both sensors synchronized in time. Centimeter level localization and manually annotated object 3D bounding boxes are also created through an off-line process. These provide accurate poses and regions-of-interest to aggregate individual LiDAR sweeps and produce dense scans that serves as our ground truth for evaluation. To ensure the quality of the aggregated point cloud, we conduct additional automatic quality assessments to remove outliers and interior points, and prune objects with poor alignment. In total, we obtain 3100 and 935 high-quality objects for training and testing respectively. Each consists of the partial LiDAR sweep within the vehicle's bounding box and the vehicle's corresponding cropped image. To remove heavily occluded vehicles, we ensure our examples have at least 100 LiDAR points.

KITTI: We also generate 209 diverse objects from 21 sequences of the KITTI tracking dataset [15]. For each object, we construct a dense ground truth 3D shape by aggregating multiple sweeps of the LiDAR data using the ground truth bounding boxes. We perform statistical filtering to reduce the noise accumulated in the aggregated shapes and manually remove noisy objects. We further symmetrize the ag-

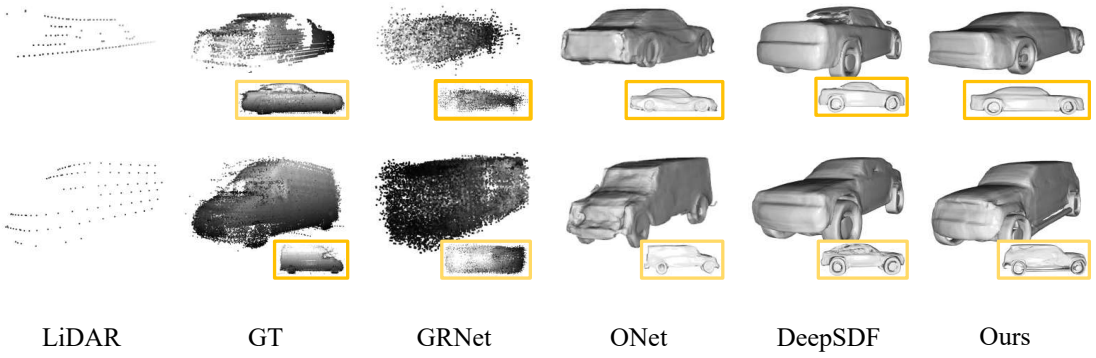


Figure 5: **LiDAR Completion Visualization on KITTI:** In comparison with others, our approach maintains high fidelity with the input point cloud and generates finer shape details even with sparse input data. However, when the input point cloud is significantly sparse, the fidelity with the GT drops. Check supp. for more visual comparison.

Method	ACD (mm) ↓	Recall (%) ↑
DIST [40]	23.40	71.99
DIST++ [40]	17.52	72.65
Ours	**5.36**	**89.05**

Table 4: **Image+LiDAR Reconstruction Results:** Combining Image and LiDAR boost the reconstruction performance of all methods. Our approach significantly outperfroms DIST and DIST++ both quantitatively and qualitatively (visual comparisons in supp.).

gregated point cloud along the lateral dimension to obtain a more complete shape. Please refer to supp. for more details.

5.2. Experimental Details

Metrics: Unlike their synthetic counterpart, real-world datasets do not possess complete watertight 3D shapes. We cannot evaluate 3D reconstruction metrics like Chamfer Distance, Volumetric IoU [36], normal consistency [36], and robust F-score [47] on them. We thus adopt asymmetric Chamfer Distance (ACD) between the aggregated ground truth point cloud and the reconstructed shape to measure the shape fidelity. ACD is defined as the sum of squared distance of each ground truth 3D point to the closest surface point on the reconstructed shape:

$$\mathrm{ACD}(\mathbf{X}, \mathbf{Y}) = \frac{1}{|\mathbf{X}|} \sum_{x \in \mathbf{X}} \min_{y \in \mathbf{Y}} \|x - y\|_2 \,.$$

We also compute the recall of the ground truth points from the reconstructed shape as a robust alternative:

$$\mathrm{Recall}(\mathbf{X}, \mathbf{Y}) = \frac{1}{|\mathbf{X}|} \sum_{x \in \mathbf{X}} \left[\min_{y \in \mathbf{Y}} \|x - y\|_2 <= t \right].$$

We set true-positive threshold $t = 0.1$ m in the paper.

Baselines: We compare against several state-of-the-art 3D object reconstruction algorithms [30, 58, 13, 32, 40]. We categorize them into two parts: (i) feed-forward methods, such as Occupancy Networks (ONet) [30] and GRNet [58]; (ii) (deep) optimization based methods, such as linear SAMP [13], DeepSDF [32], and DIST [40]. DeepSDF samples off-surface points for SDF optimization to improve robustness. We also augment the original DIST approach with such sampling procedure, referred to as DIST++.

Implementation Details: We use Adam optimizer for both training and inference. Training of our proposed approach takes 1000 epochs, 280 epochs and 360 epochs respectively for Stages 1, 2, and 3. We sample the spatial points following a grid pattern in 3D during synthetic training on Stage 1 and Stage 2; and we use aggregated LiDAR points to provide SDF supervision during the real-data training in Stage 3. During inference, in addition to using on-surface LiDAR points, we extrapolate the radial rays from the object's bounding box center to the observed suface points to generate off-surface ground truth SDF samples. In practice, we found such sampling reduces artifacts brought by LiDAR noise and missing observations. Please refer to supp. for more implementation details.

5.3. Experimental Results

LiDAR-based 3D Reconstruction on NorthAmerica: We first report LiDAR-based completion results in Tab. 1. Our approach significantly outperforms all baselines. In particular, we reduce error by 50% compared to the best deep feed-forward network GRNet and more than 25% compared to the current best deep optimization method DeepSDF. We note that SAMP has particularly large error due to its difficulty handling the larger objects such as trucks that are present in the dataset, as the embedding was trained mostly with smaller vehicles. Overall, the improve-

ments suggest the effectiveness of our approach compared to feed-forward or optimization-only based methods.

LiDAR-based Shape Reconstruction on KITTI: We compare several competing algorithms for LiDAR completion task on KITTI. Tab. 2 showcases the quantitative results. From the table we can see our method outperforms all the baselines. Note that KITTI's input points are more sparse and noisy, making the absolute errors slightly larger compared against NorthAmerica. The feed-forward GRNet performs better than optimization based methods such as DeepSDF in this case. As KITTI point clouds are more noisy this demonstrates the value of feed-forward approaches being robust. This is contrasts with reported results on the more diverse NorthAmerica dataset, where DeepSDF has better performance. This suggests the different strengths of these two types of approaches and that our method combines the power of the two.

Image-based Shape Reconstruction on NorthAmerica: Following the extension described in Sec. 4.4, we also conducted a monocular object reconstruction experiment and compared against DIST [40] which can take image alone as input to predict shape. We report the results in Tab. 3. Compared against DIST, our method produce significantly better results, especially at the invisible part, thanks to the strong shape prior induced by the deep discriminator. Note that our image-only method is competitive with the performance of DeepSDF using partial LiDAR sweeps.

Image+LiDAR 3D Reconstruction on NorthAmerica: Finally, we evaluate the performance of 3d object reconstruction that takes both image and LiDAR as input on NorthAmerica. Tab. 4 depicts the results. Our method outperfroms DIST and DIST++. In particular, compared against image or LiDAR only results, our method's use of image and LiDAR data further reduces the error by 20%.

Qualitative Results: Fig 4 compares our reconstruction results with the prior works on the NorthAmerica dataset. From this figure we could see that GRNet generates non-watertight shapes and fails to recover the fine details of car. ONet produces overly-smooth shapes at times and does not have high-fidelity to the input observation. DeepSDF maintains high-quality local details for visible regions. However, it suffers from identifying the correct global structure, thus produce the wrong shape. Our approach produces results that are both visually appealing and have high-fidelity with the input observation. Fig 5 compares our reconstruction results with the prior works on the KITTI. From these figures, we can see that our reconstructed shapes register well with the input data while generating more finer details compared to the baselines. However, when the input is very sparse, the reconstruction fails to match with the GT shape.

Enc.	Real-train	Opt.	Img.	ACD (mm) ↓	Recall (%) ↑
✓				17.2	65.89
✓		✓		7.02	86.48
✓	✓			5.96	88.80
✓	✓	✓		5.93	88.18
✓	✓		✓	8.65	84.49
✓	✓	✓	✓	**5.33**	**89.17**

Table 5: **Ablation Study:** Test-time Optimization (Opt.) and Real-train significantly boost the performance of reconstruction in the field, compared to simple feed-forward network (Row 1).

Ablation Study: In order to justify the technical components adopted in our method, we conduct a thorough ablation study on NorthAmerican dataset. In particular, we evaluate the choice of deep encoder (Enc), real-data training (Real-train), test-time deep optimization (Opt) and image as additional input (Image). Tab. 5 reports the ablation study results. From the table, we could see: 1) adding the test-time optimization significantly boost performance compared to a feed-forward network; 2) noisy/sparse real-data training boost the performance for reconstruction in the wild. 3) using image as additional input to the encoder will increase the accuracy for feed-forward encoder, and slightly improve deep optimization approach.

6. Conclusion

In this paper, we present a simple yet effective solution for 3D object reconstruction in the wild. Unlike previous approaches that suffer from sparse, partial, and potentially noisy observations, our method is able to generate accurate, high-fidelity 3D shape from real-world sensory data. The key to success is to fix the two inherent limitations of existing methods: *instability* and *low-fidelity*. Specifically, we analyze the behavior of an existing neural implicit model, DeepSDF, and discover two major defects. Based on the observations, we introduce two simple modifications to the original approach. By incorporating the adjustments into the model, we are able to significantly improve the reconstruction quality and achieve state-of-the-art performance on two real-world datasets.

References

[1] Renderpeople, 2018. 2

[2] Turbosquid. https://www.turbosquid.com. 13

[3] Abhishek Badki, Orazio Gallo, Jan Kautz, and Pradeep Sen. Meshlet priors for 3d mesh reconstruction. In *CVPR*, 2020. 2

[4] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J. Davison. Codeslam - learning a compact, optimisable representation for dense visual slam, 2019. 3

[5] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv*, 2017. 2

[6] 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, 2015. 2, 7

[7] Guanying Chen, Kai Han, and Kwan-Yee K Wong. Ps-fcn: A flexible learning framework for photometric stereo. In *ECCV*, 2018. 7

[8] Xu Chen, Zijian Dong, Jie Song, Andreas Geiger, and Otmar Hilliges. Category level object pose estimation via neural analysis-by-synthesis, 2020. 3

[9] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 2

[10] Angela Dai and Matthias Nießner. Scan2mesh: From unstructured range scans to 3d meshes. In *CVPR*, 2019. 2

[11] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *CVPR*, 2017. 2, 3

[12] Francis Engelmann, Jörg Stückler, and Bastian Leibe. Joint Object Pose Estimation and Shape Reconstruction in Urban Street Scenes Using 3D Shape Priors. In *German Conference on Pattern Recognition (GCPR)*, 2016. 2

[13] Francis Engelmann, Jorg Stuckler, and Bastian Leibe. Samp: Shape and motion priors for 4d vehicle reconstruction. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017. 2, 5, 8, 13

[14] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017. 3

[15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 2, 7

[16] R. Girdhar, D.F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 5

[17] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn, 2020. 2

[18] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation, 2018. 2

[19] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *CVPR*, 2018. 3

[20] Jiayuan Gu, Wei-Chiu Ma, Sivabalan Manivasagam, Wenyuan Zeng, Zihao Wang, Yuwen Xiong, Hao Su, and Raquel Urtasun. Weakly-supervised 3d shape completion in the wild, 2020. 2

[21] Jinjin Gu, Yujun Shen, and Bolei Zhou. Image processing using multi-code gan prior, 2020. 3, 7

[22] Swaminathan Gurumurthy and Shubham Agrawal. High fidelity semantic shape completion for point clouds using latent optimization. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019. 3

[23] Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. High-resolution shape completion using deep neural networks for global structure and local geometry inference. In *ICCV*, 2017. 2

[24] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. *ACM Trans. Graph.*, 2020. 2

[25] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *CVPR*, 2020. 3

[26] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018. 1

[27] Abhishek Kar, Shubham Tulsiani, Joao Carreira, and Jitendra Malik. Category-specific object reconstruction from a single image. In *CVPR*, 2015. 1

[28] Abhijit Kundu, Yin Li, and James M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *CVPR*, 2018. 2

[29] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. Morphing and sampling network for dense point cloud completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 2

[30] Lars M. Mescheder, Michael Oechsle, M. Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *CVPR*, 2019. 2, 3, 4, 5, 7, 8, 12

[31] Mahyar Najibi, Guangda Lai, Abhijit Kundu, Zhichao Lu, Vivek Rathod, Thomas Funkhouser, Caroline Pantofaru, David Ross, Larry S. Davis, and Alireza Fathi. Dops: Learning to detect 3d objects and predict their 3d shapes, 2020. 2

[32] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 2, 3, 4, 5, 7, 8, 12

[33] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016. 5

[34] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2, 3, 12

[35] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 2

[36] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions, 2017. 3, 8

[37] Shunsuke Saito, , Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned

implicit function for high-resolution clothed human digitization. *arXiv*, 2019. 2, 3, 4

[38] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *CVPR*, 2020. 2, 3

[39] Muhammad Sarmad, Hyunjoo Jenny Lee, and Young Min Kim. Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion. In *CVPR*, 2019. 2

[40] Songyou Peng Boxin Shi Marc Pollefeys Zhaopeng Cui Shaohui Liu, Yinda Zhang. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *CVPR*, 2020. 2, 3, 5, 7, 8, 9, 13

[41] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans, 2020. 3

[42] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *NeurIPS*, 2020. 3

[43] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *NeurIPS*, 2019. 3

[44] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017. 3

[45] Lars Mescheder Marc Pollefeys Andreas Geiger Songyou Peng, Michael Niemeyer. Convolutional occupancy networks. In *ECCV*, 2020. 2

[46] David Stutz and Andreas Geiger. Learning 3d shape completion under weak supervision. *International Journal of Computer Vision*, 2018. 2

[47] Maxim Tatarchenko, Stephan R. Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? *CoRR*, abs/1905.03678, 2019. 8

[48] Lyne P Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In *CVPR*, 2019. 2

[49] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d scene representation and rendering. *arXiv*, 2020. 2, 3

[50] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *CVPR*, 2018. 5

[51] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018. 3

[52] 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. *TOG*, 2017. 3

[53] Rui Wang, Nan Yang, Joerg Stueckler, and Daniel Cremers. Directshape: Direct photometric alignment of shape priors for visual vehicle pose and shape estimation, 2020. 2

[54] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *CVPR*, 2018. 3

[55] Xin Wen, Tianyang Li, Zhizhong Han, and Yu-Shen Liu. Point cloud completion by skip-attention network with hierarchical folding. In *CVPR*, 2020. 2

[56] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. In *NIPS*, 2017. 3, 5

[57] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 3

[58] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao, Shengping Zhang, and Wenxiu Sun. Grnet: Gridding residual network for dense point cloud completion. *arXiv*, 2020. 2, 5, 7, 8, 12

[59] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *NIPS*. 2019. 2, 3

[60] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018. 2

[61] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *3DV*, 2018. 2, 5

[62] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network, 2019. 12

[63] Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3d objects with differentiable rendering of sdf shape priors, 2020. 3

[64] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing, 2020. 3

Supplementary Material:
Secrets of 3D Implicit Object Shape Reconstruction in the Wild

Abstract

In this supplementary material, we first discuss our implementation and experimental details (Sec. A). Then we provide additional quantitative analyses in Sec. B and more qualitative results in Sec. C. We also attach a video (supp.mp4) that provides a brief overview of our method as well as animated qualitative results.

A. Additional Implementation Details

In this section, we first describe the implementation and training details about our model as well as the other baselines. Next we provide the data preparation specifics of all the datasets used in the paper.

A.1. Our Approach

Network Architecture: 1) **Decoder**: We use a pre-trained DeepSDF decoder during all the training stages. 2) **LiDAR Encoder**: We exploit the stacked PointNet proposed by Yuan *et al*. [62] as our encoder. More specifically, the first PointNet [34] block has 2 layers, with 128 and 256 units respectively. The second PointNet block has 2 layers with 512 and 1024 units. The second block takes as input both the individual point features and the global max-pooled features. The stacked PointNet encoder is followed by a final fully-connected layer (and BatchNormalization) with 256 units, with tanh as the final activation. The input to the LiDAR encoder is a partial point cloud with a maximum of 1024 points, normalized within a unit sphere using ground-truth bounding box. 3) **Image Encoder**: We use ImageNet pre-trained Resnet-18 as our backbone. It is then followed by a fully-connected layer (with 256 units) and tanh activation. 4) **Discriminator**: The architecture of our discriminator is akin to PointNet. It consists of 5 fully-connected layers (with 64, 64, 64, 128, 1024 units) for computing per-point features, and a final global max pooling layer. The input to the discriminator is a SDF volume randomly sampled from a origin-centered $(2 \times 2 \times 2)$ cube.

Training Details: The learning phase (defined in Sec. 4.2 of the main paper) is described in Fig. 6. We augment the randomly sampled points with k partial surface LiDAR points, leading to in total 4096 points. Both real and fake input SDF volumes (for GAN training) use the same 3D points. For the real SDF volumes, the SDF value corresponding to each 3D point is generated using the pre-trained latent code of the previous stage. Batch-norm is applied separately for real and fake samples. During Stage 1 and Stage 2, the SDF loss (E_{data}) is computed for 16384 randomly sampled points same as DeepSDF. For Stage 3 training on real dataset, we used aggregated ground-truth points (+ randomly jittered off-surface points) for SDF loss computation. For test-time optimization in Stage 4, we only use the partial surface Lidar points (+ randomly jittered off-surface points) for shape optimization. For all stages of training and optimization, we weighted the E_{data}, E_{prior} and E_{reg} terms in the ratio of 2:1:1.

A.2. Baselines

DeepSDF [32]: We follow the exact same implementation and point sampling guidelines as [32] to train DeepSDF on ShapeNet. In particular, we exploit 16384 SDF samples within a unit sphere enclosing the object for optimization. The shape latent code has a dimensionality of 256. The decoder is a 8 layer fully connected MLP, with ReLU as intermediate activations and tanh as the activation of the last layer. Please refer to Park *et al*. [32] for more architectural details. As for the real-world dataset, we optimize the SDF loss using the on-surface LiDAR points and the randomly sampled off-surface points. We use a maximum of 1024 on-surface points. The off-surface points are randomly sampled within a truncated distance of $\pm$ 0.02, along the rays joining the object center and the surface LiDAR points.

ONet [30]: We follow the same implementation guidelines as mentioned by the authors, except that we use 16384 occupancy samples to supervise the reconstruction of each shape during training.

GRNet [58]: We use the released codebase and the ShapeNet pre-trained weights to reconstruct shapes on KITTI and NorthAmerica datasets.

DIST/DIST++: [40] We use two camera images to optimize the 3D shape. For image-only reconstruction, we optimize the photometric loss and the latent code regularization term. For Image + LiDAR shape completion, we utilize the ground-truth depth map for additional supervision. The ground-truth sparse depth map was generated by projecting the single-sweep LiDAR points onto the camera image using ground-truth poses. For DIST++, we augment the original DIST optimization strategy with the SDF loss on off-surface points.

SAMP [13]: We first construct a PCA embedding based on 103 CAD models purchased from TurboSquid [2]. We use the purchased models instead of ShapeNet since a good PCA embedding requires shapes to be in a proper metric space and to be watertight to compute proper volumetric SDFs. Most of the purchased cad models are passenger vehicles such as sedans and mini vans. We compute volumetric SDFs for each vehicle in metric space, where the output volume has dimensions $(300 \times 100 \times 100)$. The resolution of each voxel is 0.025 meters. We set the embedding dimension to be 25, and optimize the shape embedding as well as a scaling factor on the SDF to handle larger shapes. We use Adam with learning rate 0.2. The loss function includes a smooth L1 data term, an L2 regularization term, and a scale factor regularization term. The weights of these terms are 1, 0.05, and 0.01, respectively.

A.3. Additional Details on Data Preparation

ShapeNet: We render each mesh at 5 different viewpoints to generate the sparse input point clouds required to train the LiDAR encoder in Stage 2. The transformations are randomly selected from the NorthAmerica dataset. The rendered depth maps are also sampled using the NorthAmerica's LiDAR sensor's azimuth and elevation resolutions. These sampled depth maps are then unprojected to generate the sparse point clouds.

NorthAmerica: As mentioned in the paper, NorthAmerica dataset has long trajectories of LiDAR sensor data and RGB images captured by wide-angle perspective camera. We exploit the manually annotated ground-truth detection labels to extract object-specific data from these raw sensor data. Despite using ground-truth bounding boxes, the extracted object data is still noisy (*i.e*. contains road points, non-car movable objects, etc). To reduce the noise in the extracted point cloud, we filter out the non-car points using a trained LiDAR segmentation model. The ground-truth shape is generated by aggregating multi-frame (maximum of 60 frames, captured at a rate of 10Hz) object LiDAR points in the object-coordinate space. If the object LiDAR points are less than 100 points for all frames, we discard the object. For dynamic objects, we additionally perform color-based ICP to better register the multi-sweep data.

KITTI: We use KITTI's ground-truth bounding boxes to aggregate the multi-sweep object data in the object-coordinate space. Since KITTI bounding boxes are too tight, we expand the bounding boxes by 10% along each dimension. The aggregated objects in KITTI dataset are noisier (contains interior points, flying 3D points) compared to the NorthAmerica dataset. To reduce the noise, we first perform spherical filtering by filtering out all the points, which lie at a distance greater then a specified threshold from the center of the object. Next, we apply statistical (neighbor density based) outlier removal to filter out the isolated points.

B. Quantitative Comparisons

We provide a comprehensive quantitative comparisons on the LiDAR completion task on both KITTI and NorthAmerica.

B.1. Cumulative ACD Analysis

Fig. 7a and Fig. 7b compares the cumulative ACD among various 3D shape completion approaches on KITTI and NorthAmerica datasets respectively. Our approach consistently outperforms all prior work. Compared to encoder-based intialization approaches, DeepSDF suffers from a sudden jump in the cumulative error on the KITTI dataset. We conjecture this is because some of the aggregated ground-truth objects are in fact noisy.

B.2. Recall Analysis on NorthAmerica

Fig. 8a showcase the variation of recall as a function of true-positive distance threshold. Fig. 8b analyzes the percentage of objects with recall greater than or equal to a certain value, at a distance threshold of 0.1m. Approximately, 98% of our reconstructed objects have recall $>=$ 50% and 87% of our reconstructed objects have recall $>=$ 80%, comparing to 96% and 76% of DeepSDF's object respectively.

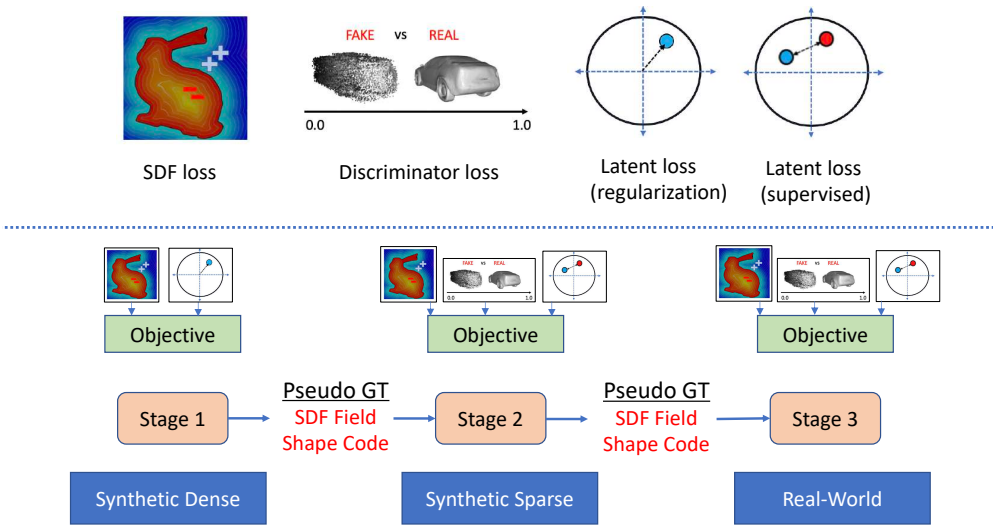
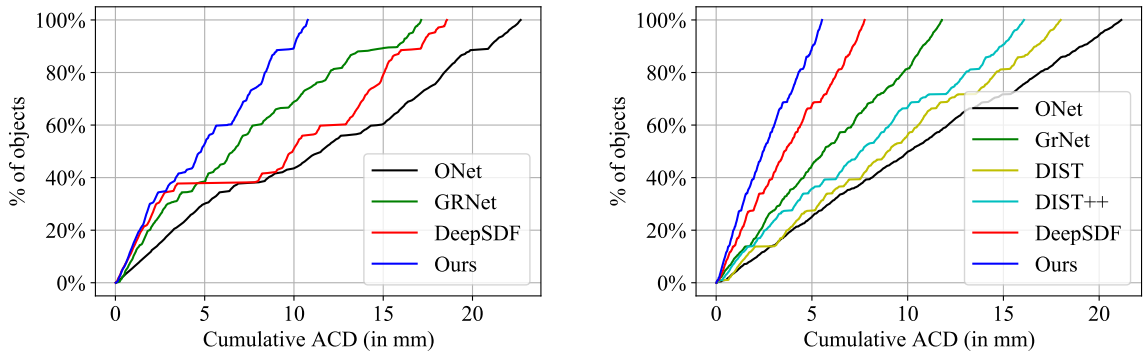


Figure 6: **Learning Phase Description:** Our learning phase consists of 3 stages: In first stage, we train the decoder module and the object shape codes on synthetic dense dataset. In second stage, we train the encoder and discriminator module (with fixed decoder module) on the synthetic sparse dataset. In third stage, we train(or finetune) encoder and discriminator modules on the real-world dataset. Output of the trained modules of each stage serve as the pseudo-GT of the following stage.

B.3. Multi-Code Optimization Analysis on NorthAmerica

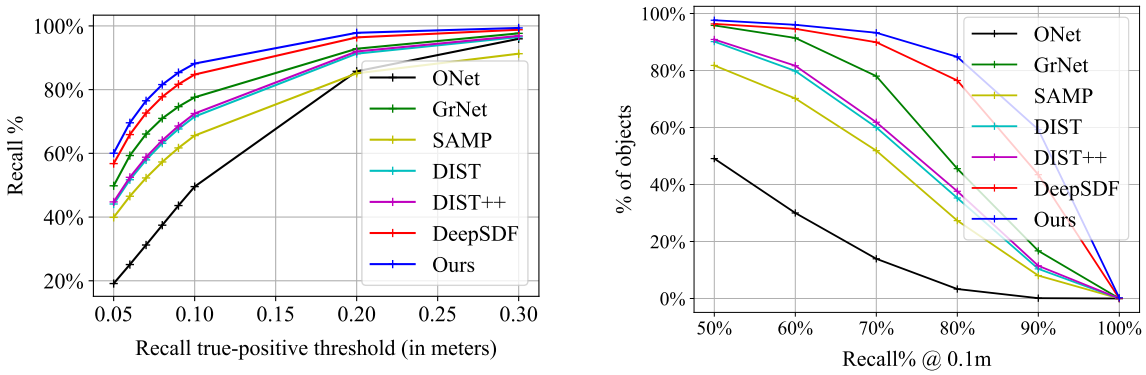
In this section, we analyse the affect of optimizing multiple shape codes instead of a single latent code. Given an initial latent code (initialized with either a random or a learned shape code), we generate n different latent codes by jittering the initial code using multiple normally-distributed noise vectors. We then jointly optimize these codes using our proposed multi-code optimization strategy, and fuse them in the same way we fused Image and LiDAR generated shape codes in the paper. Fig. 9a and Fig. 9b showcase the performance boost achieved by optimizing multiple codes. We observe boost of around 1.3-1.7% on recall, and a decrease of around 8-10% on ACD, when we optimized four latent codes instead of one, for all the experimental settings.



(a) KITTI Dataset

(b) NorthAmerica Dataset

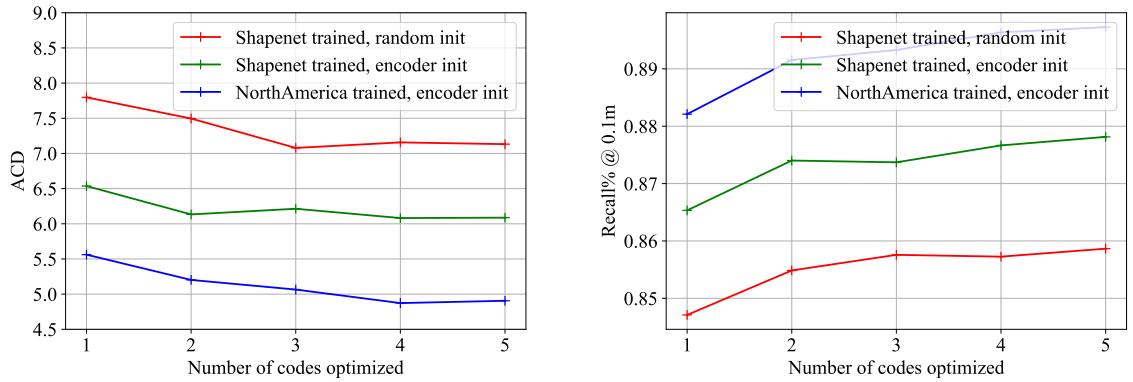
Figure 7: **Cumulative Asymmetric Chamfer Distance for LiDAR Completion:** The cumulative ACD curves showcase that our approach performs significantly better than the baselines over the entire datasets. The sharp increase in the KITTI dataset ACD metric for DeepSDF is because of the noisy objects with significant amount of outliers in the input and the GT aggregated point cloud.



(a) True-positive threshold vs Recall

(b) Cumulative Recall

Figure 8: **Recall Analysis for LiDAR Completion on NorthAmerica Dataset:** The cumulative recall curve (b.) showcases that our approach performs significantly better than the baselines over the entire NorthAmerica dataset. Moreover, as shown in curve (a.), we perform consistently better at different recall thresholds, highlighiting the greater fidelity and robustness achieved by our approach.



(a) ACD on NorthAmerica

(b) Recall on NorthAmerica

Figure 9: **Performance vs # of Codes in Multi-Code Optimization:** Multi-code optimization boosts the performance of all approaches irrespective of the latent code initialization and the training data used.

C. Qualitative Visualizations

C.1. LiDAR Shape Completion

We show additional LiDAR completion results on NorthAmerica and KITTI datasets in Fig. 10 and Fig. 11 respectively. Green points depict the overlay of the GT point clouds. While GRNet fails in completing the full shape, ONet tends to predict over-smooth shapes. Both DeepSDF and our approach produce completed and high-quality shapes, yet our reconstructed meshes have more fine-grained, structured details.

C.2. Monocular Image Reconstruction

Fig. 12 compares Image-based 3D reconstruction approaches on NorthAmerica dataset. While DIST/DIST++ use two images (either the stereo pair, or images captured at consecutive frames) to optimize the 3D shape, we use only one single image to generate the shape latent code. Our results yet still have higher fidelity than those of DIST. The last row shows an example where our encoder-only method produces a comparatively wrong shape.

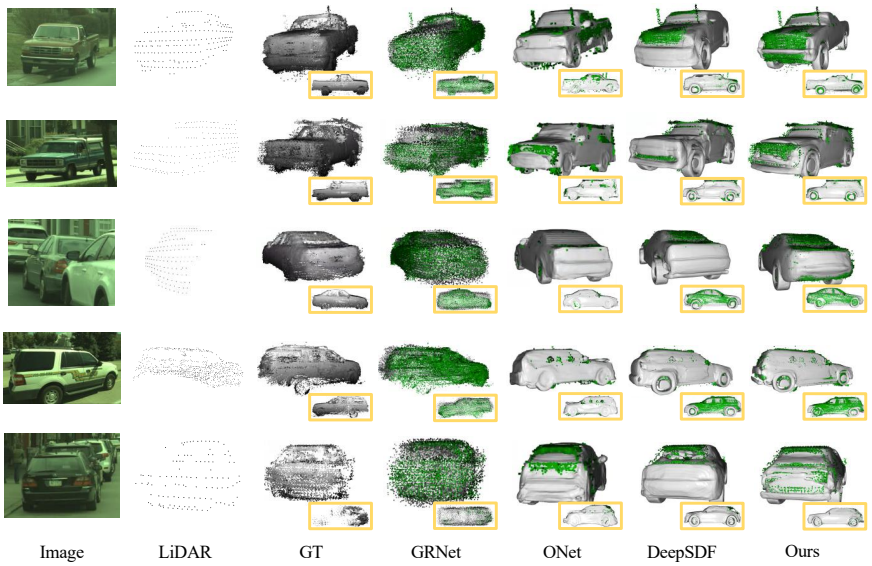


Figure 10: **LiDAR Completion on NorthAmerica:** Our approach combines the strength of both feed-forward and optimization based approaches and showcases significant improvements over the prior works in terms of learning better global shape, robustness to input noise, better regsitration with the GT and in maintaining finer details.

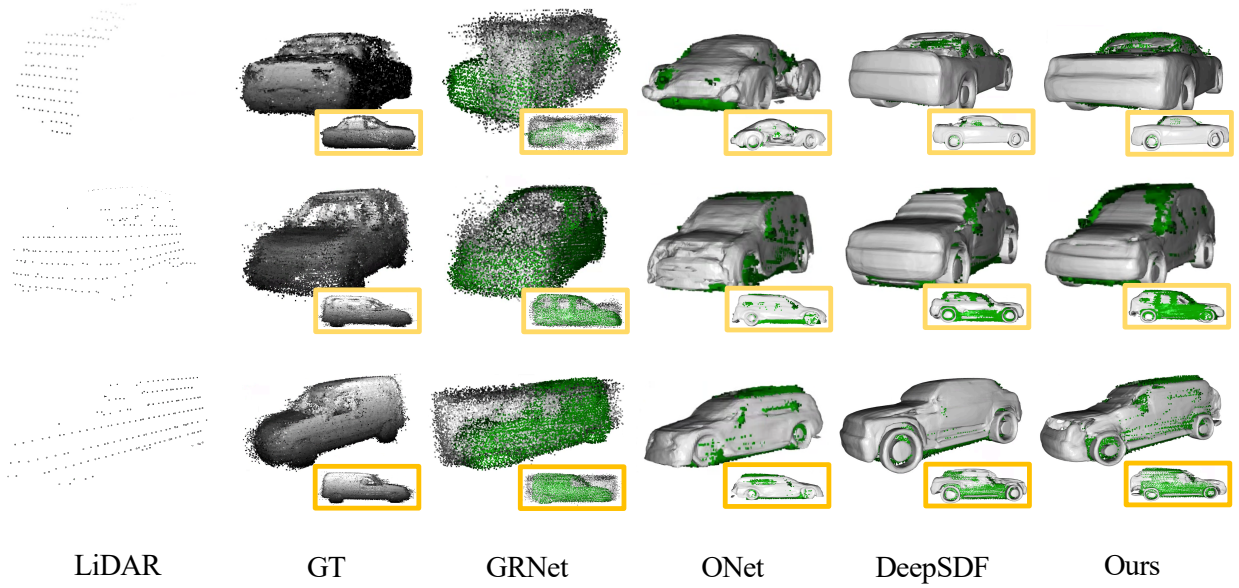


Figure 11: **LiDAR Completion on KITTI:** The registration results showcase the efficiency of our approach even on the noisy KITTI dataset.

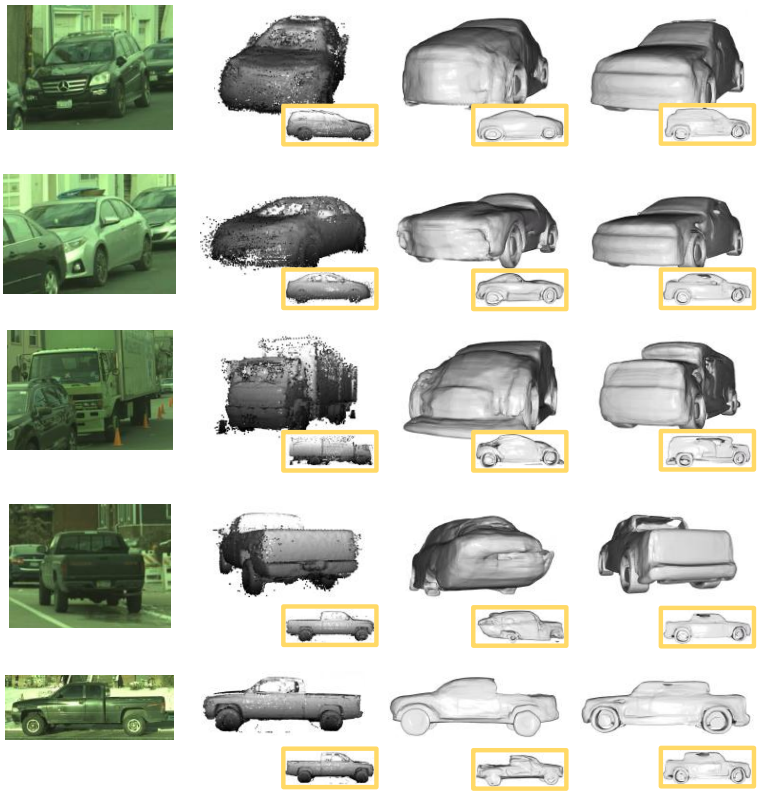


Figure 12: **Image-based Reconstruction on NorthAmerica:** Compared to DIST which uses multiple images, our approach uses just a single image. Despite that, our results still have higher fidelity than those of DIST. The last row shows an example where our encoder-only method produces a comparatively wrong shape.

C.3. Image + LiDAR Shape Completion

Fig. 13 compares Image + LiDAR Shape Completion approaches on NorthAmerica. Adding LiDAR to image-only reconstruction pipelines helps all the approaches (DIST/ DIST++/ Ours) in generating shapes with higher fidelity. Thanks to the proposed neural initialization/ regularized optimization, our method's predicted shapes align well with GT shape and maintain much finer details.

C.4. Ablation Study on Image + LiDAR 3D Reconstruction on NorthAmerica

Fig. 14 visually compares the shapes reconstructed using single image, single LiDAR sweep and single image + single LiDAR sweep together. Monocular Image Reconstruction is a feed-forward approach, where a single image is used to reconstruct the 3D shape. On the other hand, LiDAR and Image + LiDAR completion approaches iteratively optimize the 3D shape at test-time as proposed in the paper. As can be seen in the figure, shapes reconstructed using monocular image have high fidelity, pertaining to the proposed multi-stage training regime. Further, adding the test-time optimization significantly boost performance compared to a feed-forward network, making the generated shapes register well with the ground-truth shape.

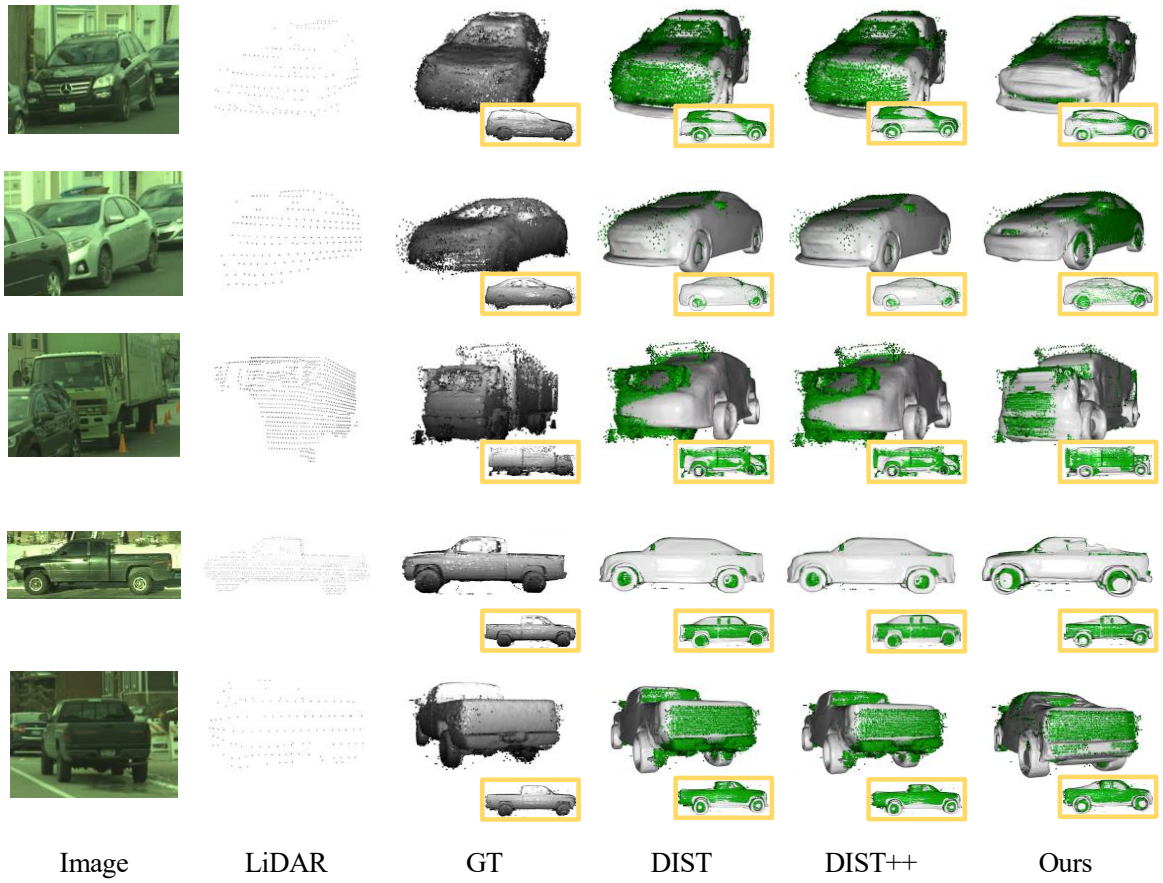


Figure 13: **Image+LiDAR Shape Completion on NorthAmerica:** Compared to DIST/ DIST++, our approach maintains both the global structure and the finer dettails much better.

Figure 14: **Image+LiDAR Ablation on NorthAmerica:** Thanks to the multi-stage training regime, the monocular-image ablation generates higher-fidelity results. Test-time optimization using 3D LiDAR further boosts the fidelity performance. The last row showcases an example, where reconstruction from monocular image is ambiguous because of the pose of the object. Utilizing the 3D information helps resolve the ambiguity.