Supplementary Material
Mending Neural Implicit Modeling for 3D Vehicle 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] Raquel Urtasun[1,2]
[1]Uber ATG, [2]University of Toronto, [3]Massachusetts Institute of Technology

1. Abstract

In this supplementary material, we first provide additional experimental details, which include dataset preparation details, implementation details and baselines' descriptions. (Sec. 2). Then, we provide additional experimental results in Sec. 3 covering both quantitative and qualitative analysis for the three shape reconstruction tasks, we defined in the paper: LiDAR-based Shape Completion (Sec. 3.1), Monocular Image Reconstruction (Sec. 3.2) and Image + LiDAR Shape Completion (Sec. 3.3). We also attach a video (supp.mp4) that provides a brief overview of our method as well as animated qualitative results.

2. Additional Experimental Details

In this section, we first provide additional data preparation details for all three datasets (ShapeNet, NorthAmerica, KITTI). Next, we describe the implementation and training details for our model as well as the baselines for the in-the-wild LiDAR shape completion task.

2.1. Dataset Details

ShapeNet: We utilize 2364 shapenet meshes for training our shape reconstruction pipeline. To generate the ground-truth SDF volumes for supervising the training stages (stage 1 and stage 2), we first normalize each mesh within a unit sphere and then randomly sample 16384 points from the sphere. For each sampled point, GT SDF value is generated by measuring its distance from the mesh surface along the GT surface normal. We also render each mesh at 5 different viewpoints to generate the sparse input point clouds required to train the LiDAR encoder in Stage 2. The viewing transformations used for rendering are randomly selected poses 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. We used Blender's cycles engine to render the meshes.

NorthAmerica: NorthAmerica dataset has long trajectories of sparse point clouds captured by a 64-line spinning 10Hz LiDAR sensor and RGB images captured by a wide-angle perspective camera, with both the 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 serve as our GT for evaluation. Specifically, we exploit these 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 [14]. 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. We further symmetrize the aggregated point cloud along the lateral dimension to create a more complete GT shape. For dynamic objects, we additionally perform color-based ICP [9] (using LiDAR intensity values) to better register the multi-sweep data.

*Equal Contribution.
‡Work done as part of internship.

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, which causes parts of the object to not be included during aggregation, 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 than a specified threshold from the center of the object. Next, we apply statistical (neighbor density based) outlier removal to filter out the isolated points. Like NorthAmerica dataset, we symmetrize the aggregated point cloud along the lateral dimension.

2.2. Implementation Details

Curriculum Training and Inference procedure: 1) **Training Stage 1:** Our shape reconstruction pipeline consists of encoder, decoder and discriminator module. As described in the main paper, we use a 2-stage curriculum learning strategy to train our model on the synthetic Shapenet dataset. In the first stage, similar to DeepSDF[10], we train an auto-decoder framework to jointly optimize the (256-dimensional) shape latent-code and the decoder weights. The loss function for stage 1 training is: $\mathcal{L}^{\text{dec}} + \mathcal{L}^{\text{reg}}$, where $\mathcal{L}^{\text{dec}}$ loss term is supervised by the ground-truth SDF volumes (with 16384 randomly sampled points). We train the auto-decoder framework for 1000 epochs in Stage 1. 2) **Training Stage 2:** Next, we train the encoder and the discriminator (keeping the decoder fixed) for 280 epochs in stage 2. The input to the LiDAR encoder is a partial point-cloud with *k* 3D points ($k \leq 1024$). The encoder then outputs a 256-dimensional latent-code. Given this latent-code and a grid of 3D points, the decoder predicts the corresponding SDF volume. The discriminator then classifies the predicted SDF volume as real or fake. For training the discriminator, we first generate real and fake SDF volumes. Both real and fake SDF volumes use the same set of 3D points. These 3D points are randomly sampled from the whole 3D space and are augmented with the *k* input points, leading to a total of 4096 points. For the real samples, the SDF values corresponding to the 3D points are generated using stage 1's pre-trained latent-codes, while for the fake SDF volumes we use the encoder predicted latent-code. The loss function for stage 2 training is: $\mathcal{L}^{\text{dec}} + \mathcal{L}^{z} + \mathcal{L}^{\text{gan}}$, where $\mathcal{L}^{\text{dec}}$ is same as stage 1's $\mathcal{L}^{\text{dec}}$ loss term. $\mathcal{L}^{z}$ loss term is supervised by the trained latent-codes of stage 1. $\mathcal{L}^{\text{gan}}$ loss term which uses the real and fake SDF volumes, guides the optimization of discriminator weights. For both the training stages, we weight the $\mathcal{L}_{\text{dec}}$, $\mathcal{L}_{\text{gan}}$ and $\mathcal{L}_{\text{reg}}$(or $\mathcal{L}_{\text{z}}$) terms in the ratio of 2:1:1. 3) **Inference:** Finally, at test-time, given in-the-wild real-world observations, we utilize the trained encoder module to predict an initial latent-code. This latent code is then optimized for 800 iterations using the inference procedure defined in the paper Sec. 4.2. We use first-order gradient optimizer [6] for both training and inference. During inference also, we weight E_{data}, E_{reg} and E_{dis} in the ratio 2:1:1.

Architecture Details: 1) **Decoder**: The decoder architecture is exactly same as DeepSDF [10]. 2) **LiDAR Encoder**: We exploit the stacked PointNet network proposed by Yuan *et al.* [13] as our encoder. More specifically, the first PointNet [11] 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. 3) **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. We separately compute batch-norm [5] mean and variance statistics for the real and the fake samples.

2.3. Baselines

DeepSDF [10]: We follow the exact same implementation and point sampling guidelines as [10] 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 an 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.* [10] 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 [8]: 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 [12]: We use the released codebase and the ShapeNet pre-trained weights to reconstruct shapes on KITTI and NorthAmerica datasets.

DIST/DIST++: [7] For LiDAR completion task, DIST requires ground-truth depth maps for supervising their auto-decoder optimization framework. 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 [3]: We first construct a PCA embedding based on 103 CAD models purchased from TurboSquid [1]. 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.

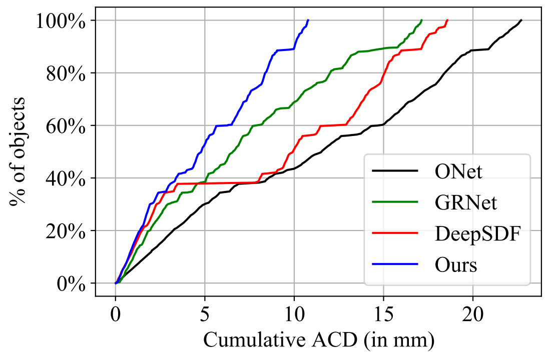
3. Additional Experimental Results

3.1. LiDAR-based Shape Completion

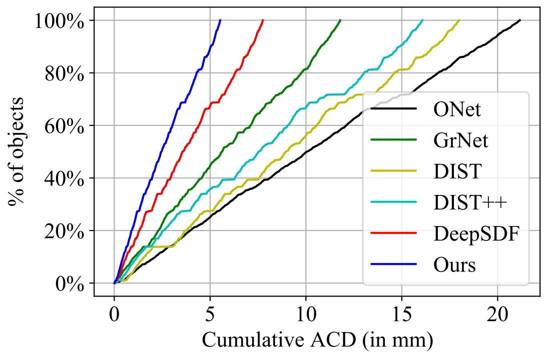
In this section, we compare all the LiDAR-based shape completion approaches both quantitatively (Cumulative ACD and Recall comparison) and qualitatively.

Cumulative ACD Analysis: Fig. 1a and Fig. 1b 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 initialization 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.

Recall Analysis on NorthAmerica: Fig. 2a showcase the variation of recall as a function of true-positive distance threshold. Fig. 2b 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.



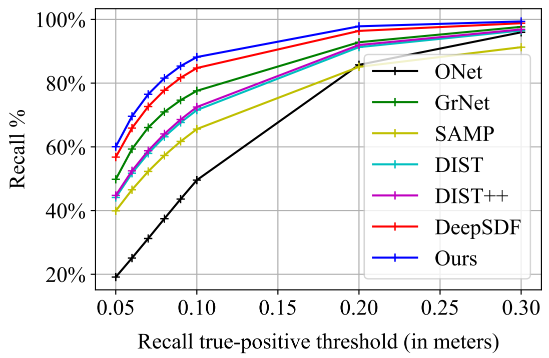
(a) KITTI Dataset



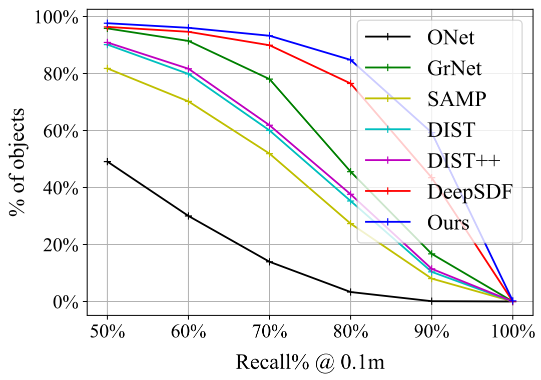
(b) NorthAmerica Dataset

Figure 1: Cumulative Asymmetric Chamfer Distance for LiDAR Completion

Qualitative Analysis: Fig. 6 and Fig. 4 compares Ours$_{\text{no-finetune}}$ results with the prior works on NorthAmerica and KITTI dataset respectively. No fine-tuning on real-world datasets was done for these visualizations. Green points depict the overlay of the GT point clouds. As shown in Fig. 6, our approach creates much cleaner shapes, maintaining both the global structure and the fine details of the GT shape better than the state-of-the-art DeepSDF[10] method. Fig. 5 further compares Ours$_{\text{finetune}}$ results (obtained by fine-tuning the ShapeNet pre-trained model on NorthAmerica dataset) with the prior works.



(a) True-positive threshold vs Recall



(b) Cumulative Recall

Figure 2: Recall Analysis for LiDAR Completion on NorthAmerica Dataset

While GRNet fails in completing the full shape, ONet tends to predict over-smooth shapes. Both DeepSDF and our approach produce complete and high-quality shapes, yet our reconstructed meshes have more fine-grained, structured details.

3.2. Monocular Image Shape Reconstruction

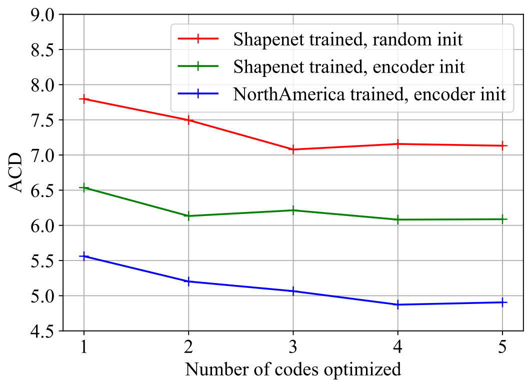
In this section, we first describe the training procedure used by our single-image 3D reconstruction approach. Next, we showcase some qualitative comparisons between our image reconstruction method and the state-of-the-art DIST[7] approach.

Training procedure: Similar to our LiDAR completion model, our image reconstruction pipeline consists of an image encoder, decoder and discriminator. As mentioned in the paper Sec. 6.2, we directly train our image encoder and discriminator on the real-world NorthAmerica dataset (training set of 3100 instances). Prior to training the image encoder, we first train our LiDAR completion model (LiDAR encoder, decoder and discriminator) on the Shapenet dataset as mentioned in Sec. 2.2. We then utilize this pre-trained model and perform inference using real-world LiDAR observations to generate a shape latent-code (predicted by the LiDAR encoder) and a SDF volume (predicted by the decoder for a randomly sampled grid of 3D points). The predicted shape latent-codes serve as the pseudo-GT for supervising the training of the image encoder, and the predicted SDF volumes serve as the real input samples for training the discriminator. In addition to the pseudo-GT, we have sparse on-surface LiDAR points (with SDF=0) which serve as the GT SDF data. Rest, following the training procedure mentioned in paper Sec. 4.3, we train the image encoder and discriminator for 360 epochs on NorthAmerica dataset training set.

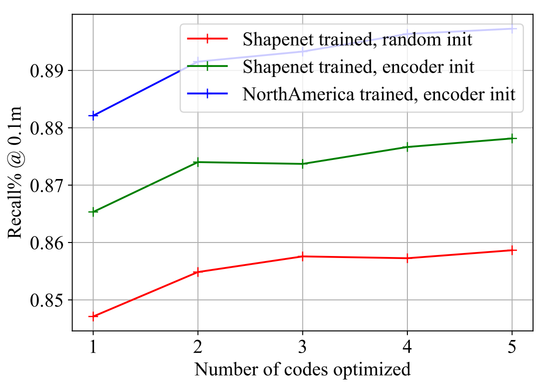
Qualitative Analysis: Fig. 7 compares image-based 3D reconstruction approaches on NorthAmerica dataset. For DIST[7], we use two camera images to optimize the 3D shape and we optimize the photometric loss and the latent code regularization term during test-time optimization of the latent-code. Unlike DIST, we use only one single image to generate the shape latent code. Our results yet still have significantly higher fidelity than those of DIST. Since the image reconstruction model is trained using noisier real-world LiDAR data and the captured LiDAR is particularly more noisy near the transparent windows, we notice some holes near the vehicle windows in some of the reconstructed shapes. Please check the visual comparison in the attached video for multi-view (gif-based) visualization of the reconstructed shapes.

3.3. Image + LiDAR Shape Completion

As mentioned in the paper Sec. 6.3, our approach allows us to combine multiple sensor modalities for shape reconstruction without any re-training. We utilize a multi-code fusion technique to combine the image and LiDAR observations directly at test-time. In this section, we first provide a detailed description of the multi-code fusion and optimization technique, followed by its quantitative analysis. Finally, we showcase some qualitative comparisons and ablations for image + LiDAR 3D reconstruction.



(a) ACD on NorthAmerica



(b) Recall on NorthAmerica

Figure 3: Performance vs # of codes in multi-code optimization

Multi-code optimization: Given Image and LiDAR observations, we simply use the two pre-trained encoder modules (point-cloud encoder and image-encoder) and generate two shape latent-codes, $\mathbf{z}_{\text{img}}^{\text{init}}$ and $\mathbf{z}_{\text{LiDAR}}^{\text{init}}$. Next, inspired by the latest effort on GAN inversion [4] and photometric stereo [2], we propose to fuse the two latent codes at the feature level. We first divide decoder module ($\mathbf{f}_\theta$) into two sub-networks $\mathbf{f}_{\theta_1}^l$ and $\mathbf{f}_{\theta_2}^l$, where l indicates the layer that we split, θ_1 and θ_2 are parameters of the two sub-networks and $\mathbf{f}_\theta = \mathbf{f}_{\theta_2}^l \circ \mathbf{f}_{\theta_1}^l$. Then we pass both latent codes into the first sub-network, $\mathbf{f}_{\theta_1}^l$, and generate two corresponding feature maps. Finally the estimated feature maps are aggregated and passed into the second sub-network $\mathbf{f}_{\theta_2}^l$. Using this multi-code fusion procedure, the inference procedure becomes:

$$\begin{aligned}\mathbf{z}^* = \underset{\mathbf{z}_{\text{img}},\mathbf{z}_{\text{LiDAR}}}{\operatorname{argmin}}\ & E_{\text{data}}(\mathbf{o}, \mathbf{f}_{\theta_2}^l(\mathbf{X}, \mathbf{z}_l)) + \lambda_{\text{reg}} E_{\text{reg}}(\mathbf{z}_{\text{img}}) \\ & + \lambda_{\text{reg}} E_{\text{reg}}(\mathbf{z}_{\text{LiDAR}}) + \lambda_{\text{dis}} E_{\text{dis}}(\mathbf{f}_{\theta_2}^l(\mathbf{X}, \mathbf{z}_l)),\end{aligned} \tag{1}$$

where $\mathbf{z}_l = g(\mathbf{f}_{\theta_1}^l(\mathbf{x}, \mathbf{z}_{\text{img}}),\ \mathbf{f}_{\theta_1}^l(\mathbf{x}, \mathbf{z}_{\text{LiDAR}}))$ is the aggregated feature and g is the aggregation function. $\mathbf{z}_{\text{LiDAR}}$ and $\mathbf{z}_{\text{img}}$ are initialized using $\mathbf{z}_{\text{LiDAR}}^{\text{init}}$ and $\mathbf{z}_{\text{img}}^{\text{init}}$ respectively. E_{data}, E_{reg} and E_{dis} are same as defined in paper Sec. 4.2. Following [2], we adopt max-pooling to fuse the features. The advantages of the proposed multi-fusion technique are three fold: (i) there are no extra parameters required for multi-sensor fusion; (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.

Quantitative Analysis: The proposed multi-code optimization technique can even be used with a single sensor observation. Given an observation (eg: an image or a LiDAR point cloud), we first generate an initial latent code. Then, 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. The same multi-code optimization procedure can be performed for DeepSDF approach, where instead of one, we optimize n different random latent-codes using the auto-decoder framework. Fig. 3a and Fig. 3b showcase the performance boost achieved by optimizing multiple codes, which is irrespective of the latent-code initialization and the training dataset used. We observe a 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.

Qualitative Comparison: Fig. 8 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.

Ablation Study on Image + LiDAR 3D Reconstruction on NorthAmerica: Fig. 9 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 boosts performance compared to a feed-forward network, making the generated shapes register well with the ground-truth shape.

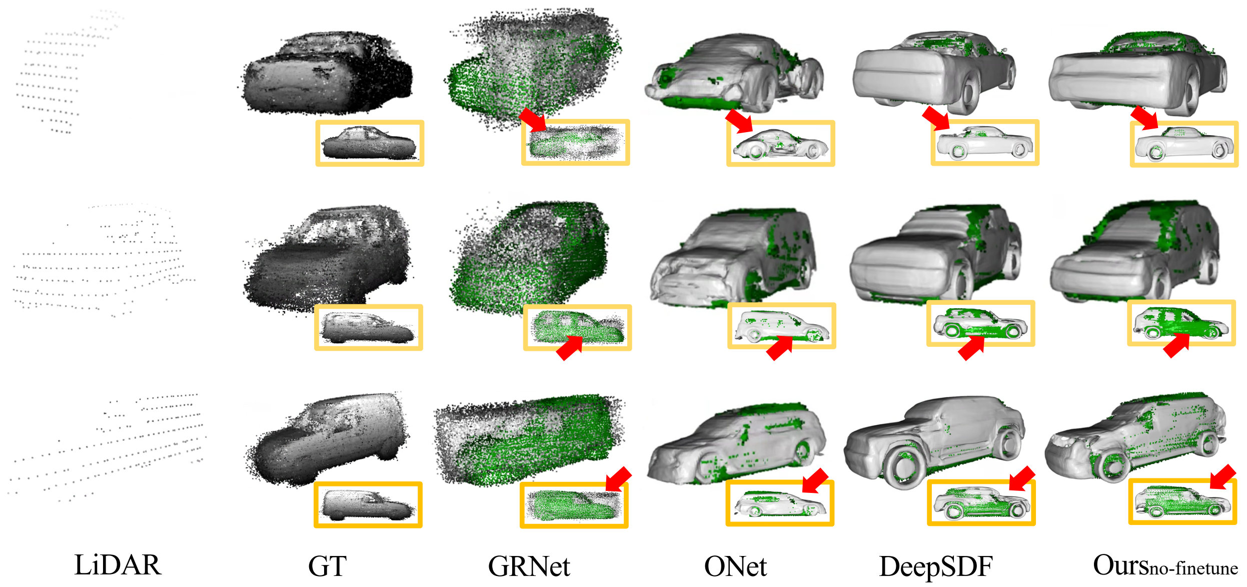


Figure 4: **LiDAR Completion on KITTI:** In the KITTI dataset, LiDAR point clouds are much noisy and sparse, and the vehicle bounding boxes needed to normalize the input data are not accurate either. Because of this all KITTI reconstructions have more artifacts compared to NorthAmerica reconstructions. Compared to all the other approaches, our shapes have much less artifacts and maintain finer details.

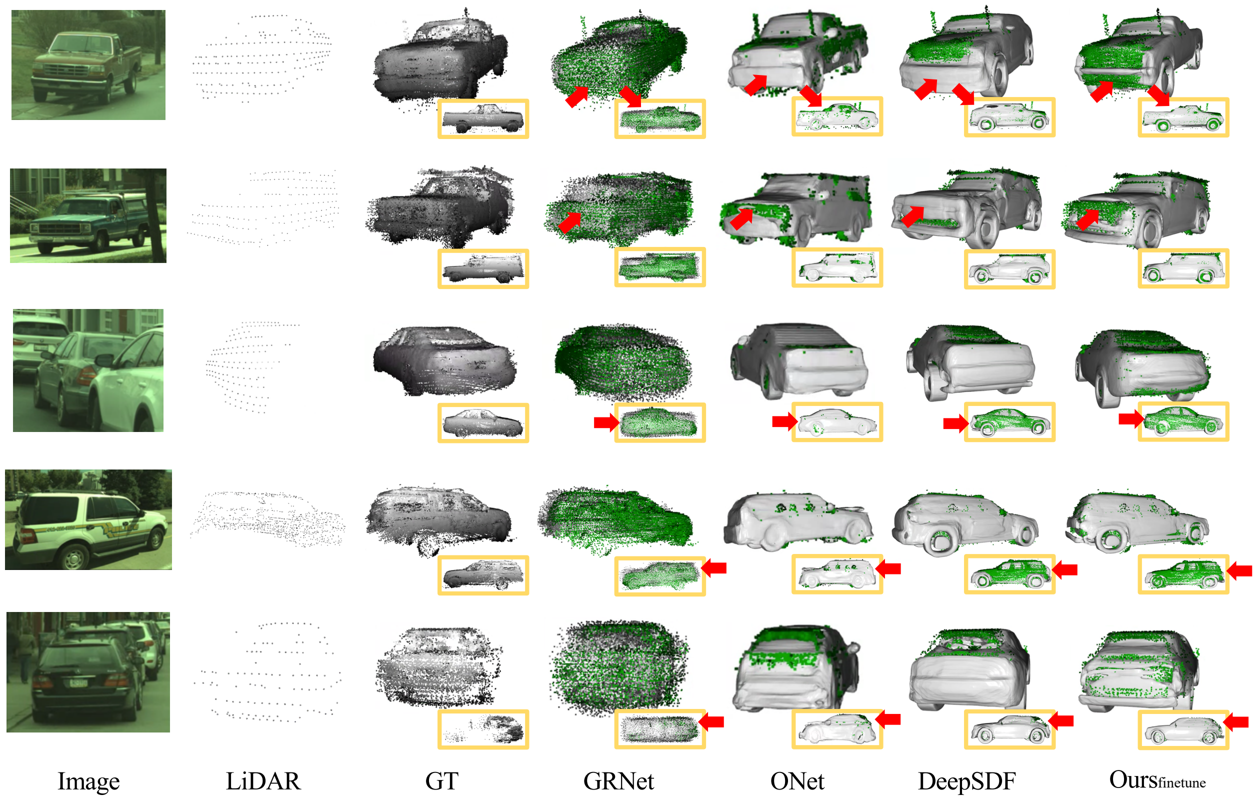


Figure 5: **LiDAR Completion on NorthAmerica:** Compared to prior works, our approach (1) maintains fine details, (2) register well with the input and the GT and, (3) works well with sparse and occluded observations. Please zoom in for enhanced visual comparison.

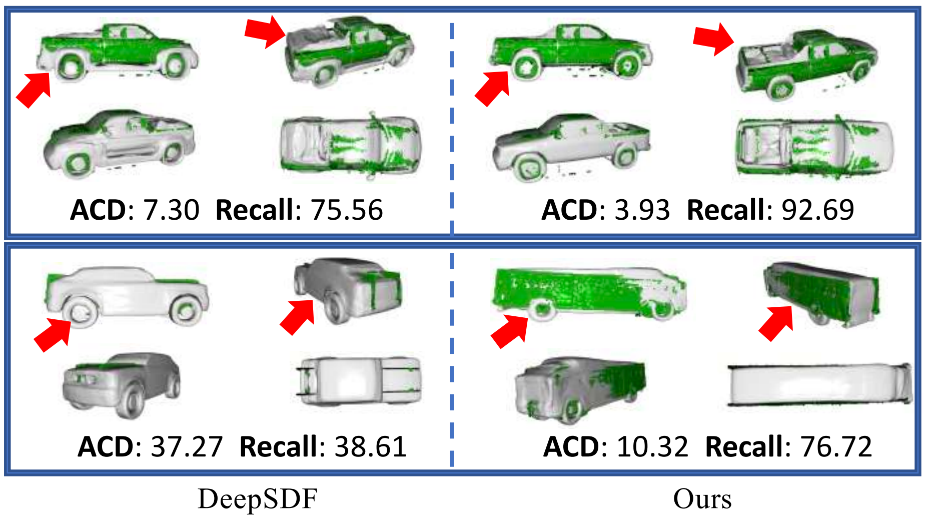


Figure 6: **In-the-wild LiDAR Completion Comparison (Ours$_{\text{no-finetune}}$ vs DeepSDF) on NorthAmerica.** Our approach generates much cleaner shapes and maintains the fine details even without any real-world fine-tuning (eg: compare the cavity of the reconstructed pickup cars in Row 1). Our proposed framework is also generalizable to unseen vehicles (eg: check the unseen NorthAmerica dataset bus in Row 2).

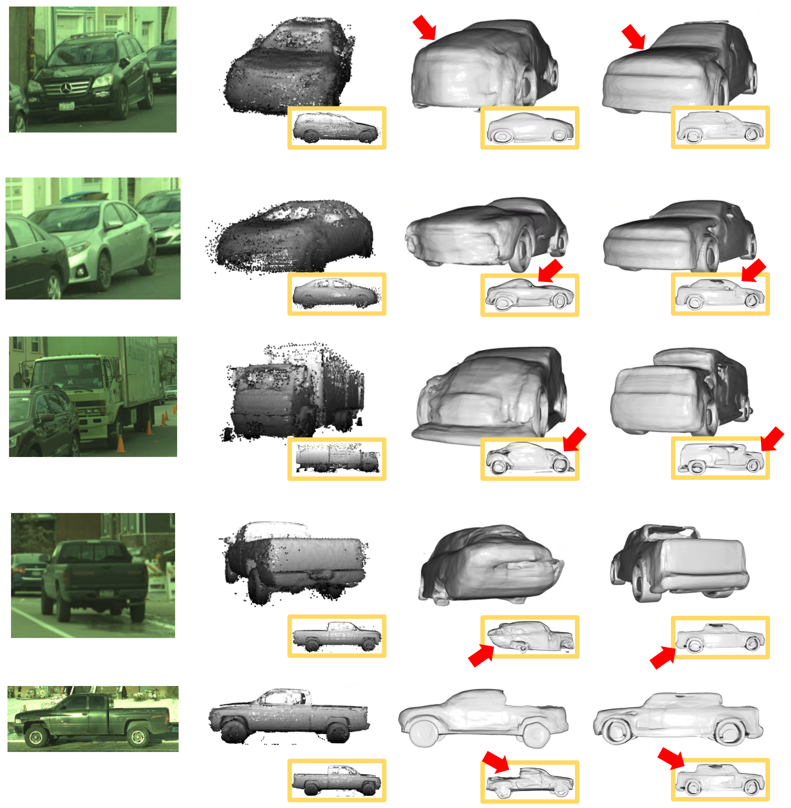


Figure 7: **Image-based Reconstruction on NorthAmerica:** Thanks to the learned shape priors, we can easily train our model on real-world datasets using only sparse on-surface points as GT. Compared to DIST, our approach has much less artifacts and performs well even in the regions occluded in the input image. Check the gif visualizations in the attached video for multi-view comparson of the reconstructed shapes.

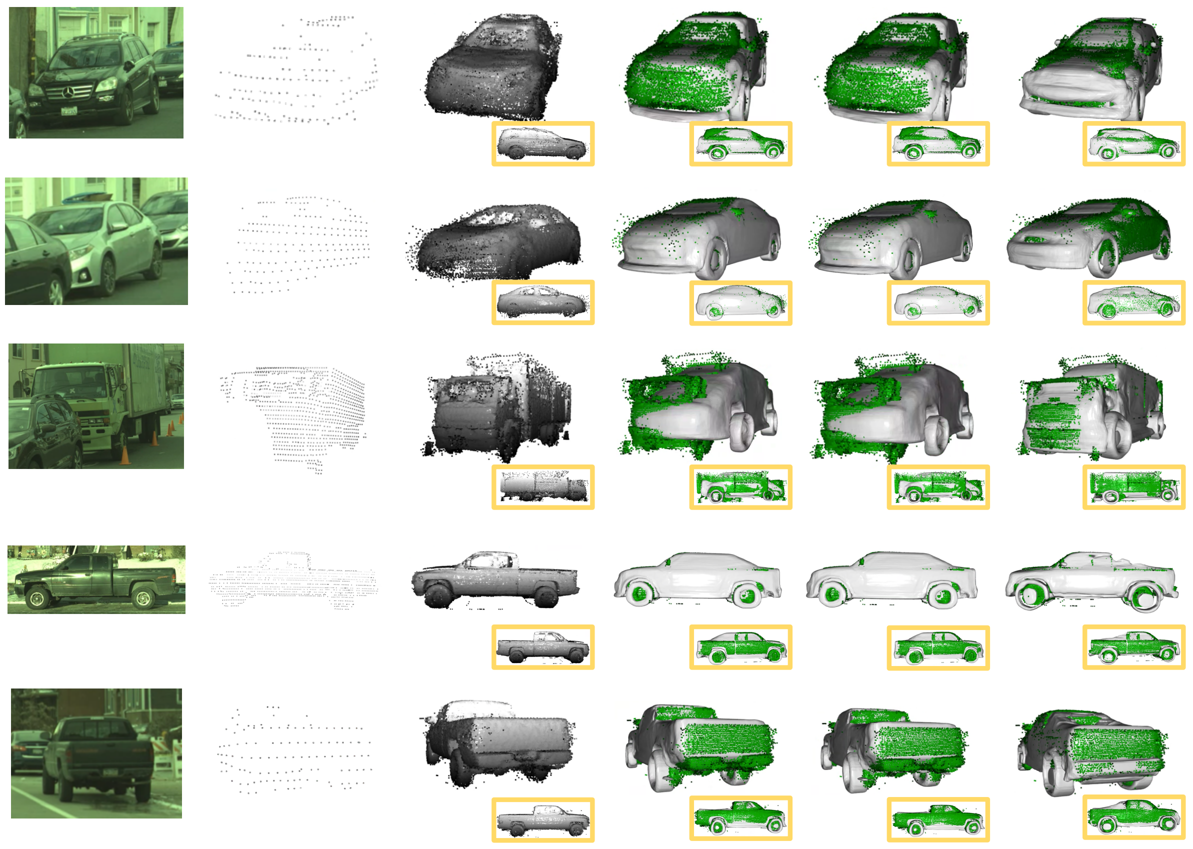


Figure 8: **Image+LiDAR Shape Completion on NorthAmerica**

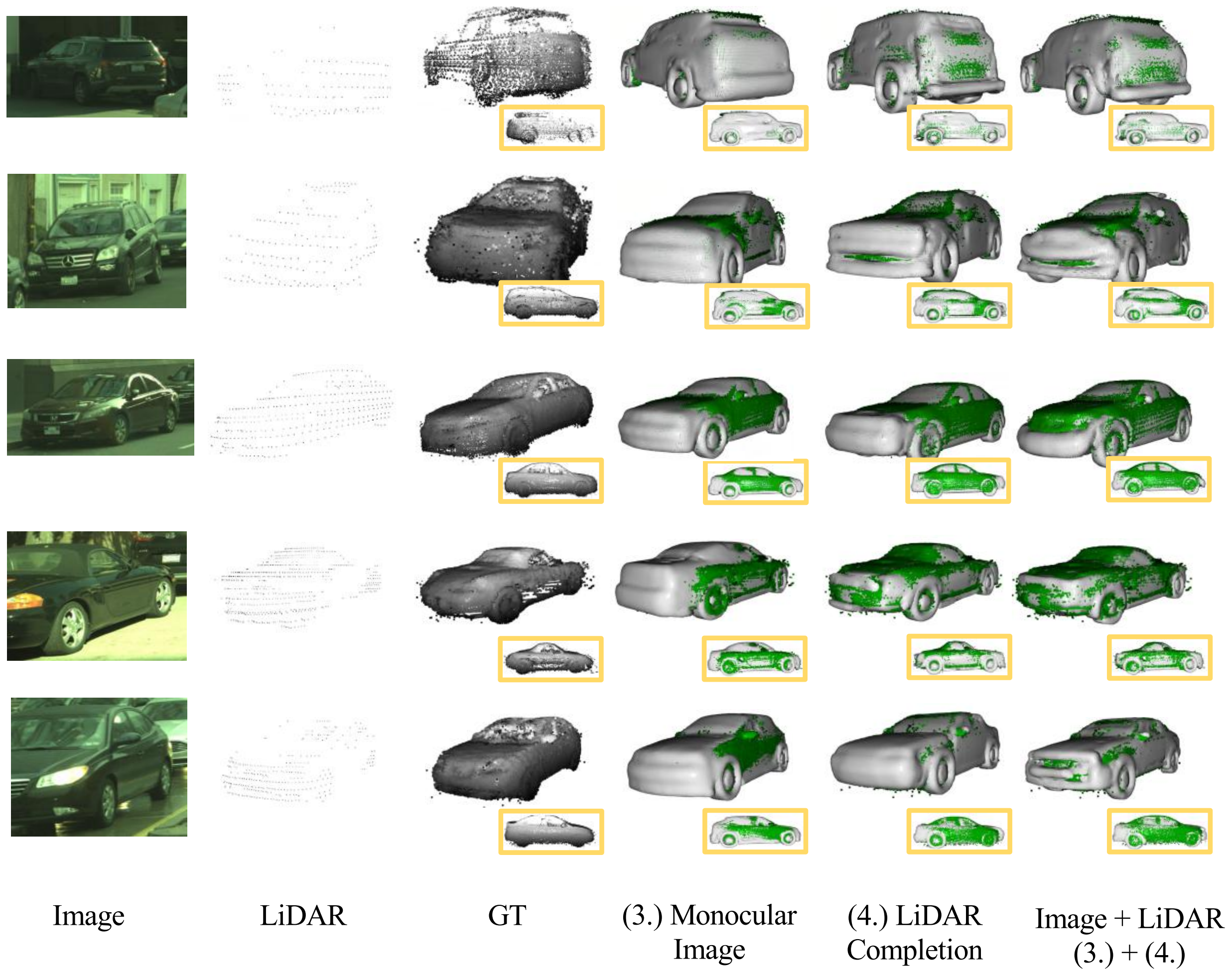


Figure 9: **Image+LiDAR ablation on NorthAmerica**

References

[1] Turbosquid.

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

[3] Francis Engelmann, Jorg Stuckler, and Bastian Leibe. Samp: Shape and motion priors for 4d vehicle reconstruction. *WACV*, 2017.

[4] Jinjin Gu, Yujun Shen, and Bolei Zhou. Image processing using multi-code gan prior. In *CVPR*, 2020.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

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

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

[9] J. Park, Q. Zhou, and V. Koltun. Colored point cloud registration revisited. In *ICCV*, 2017.

[10] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.

[11] 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.

[12] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao, Shengping Zhang, and Wenxiu Sun. Grnet: Gridding residual network for dense point cloud completion. In *ECCV*, 2020.

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

[14] C. Zhang, W. Luo, and R. Urtasun. Efficient convolutions for real-time semantic segmentation of 3d point clouds. In *3DV*, 2018.