2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

NF-SLAM: Effective, Normalizing Flow-supported Neural Field representations for object-level visual SLAM in automotive applications

Li Cui[1], Yang Ding[1], Richard Hartley[2], Zirui Xie[1], Laurent Kneip[3,1] and Zhenghua Yu[1]

***Abstract*— We propose a novel, vision-only object-level SLAM framework for automotive applications representing 3D shapes by implicit signed distance functions. Our key innovation consists of augmenting the standard neural representation by a normalizing flow network. As a result, achieving strong representation power on the specific class of road vehicles is made possible by compact networks with only 16-dimensional latent codes. Furthermore, the newly proposed architecture exhibits a significant performance improvement in the presence of only sparse and noisy data, which is demonstrated through comparative experiments on synthetic data. The module is embedded into the back-end of a stereo-vision based framework for joint, incremental shape optimization. The loss function is given by a combination of a sparse 3D point-based SDF loss, a sparse rendering loss, and a semantic mask-based silhouette-consistency term. We furthermore leverage semantic information to determine keypoint extraction density in the front-end. Finally, experimental results on real-world data reveal accurate and reliable performance comparable to alternative frameworks that make use of direct depth readings. The proposed method performs well with only sparse 3D points obtained from bundle adjustment, and eventually continues to deliver stable results even under exclusive use of the mask-consistency term.**

I. INTRODUCTION

Beyond merely estimating ego-motion and environment geometry, we require Simultaneous Localization And Mapping (SLAM) algorithms for automotive applications to generate an object-level understanding of the scene and estimate the dynamics and shape of surrounding vehicles. In line with recent works such as DSP-SLAM [1], we propose a novel vision-only object-level SLAM framework able to estimate the exact pose and shape of nearby vehicles.

The dominating shape representation embedded into the back-end optimization of object-level SLAM frameworks is currently given by implicit neural shape generators such as DeepSDF [2]. The network consists of a Multi-Layer Perceptron (MLP) conditioned by an optimizable latent code to predict the distance to the surface for an input sampling point. The latter may be chosen continuously in space, thereby adding a lot of flexibility to the sampling scheme and the way in which such representations can be embedded into the back-propagating optimization thread of both vision-based frameworks such as DSP-SLAM [1], or lidar-based frameworks such as TwistSLAM++ [3]. However, as pointed out by Duggal et al. [4], the raw DeepSDF representation faces difficulties in the presence of sparse or noisy visual measurements, and also returns strongly varying optimization results for only small variations of the initial latent code. It comes as no surprise that in DSP-SLAM [1]—a framework that can be used both with and without lidar measurements—vision-only results perform vastly inferior to their counterpart that makes use of direct and denser depth readings.

In light of these difficulties, we propose NF-SLAM, a novel object-level stereo visual SLAM framework that incorporates the following innovations:

- A sparse front-end feature detector with an adaptively tuned extraction density making use of semantic segmentation to identify regions of interest (i.e., vehicle detections). While the output remains sparse, this step enriches the object-related 3D point cloud that can be obtained from bundle adjustment.
- A novel generative shape representation that augments DeepSDF by a normalizing flow network, thereby stabilizing and improving results obtained from sparse visual measurements. The representation is readily embedded into optimization and makes use of only 16-dimensional latent codes as opposed to the 64-dimensional codes used in DSP-SLAM [1].
- A complete object-level SLAM framework that successfully marries these novel front and back-end modules with an incremental optimization objective. The latter makes use of an SDF loss, a sparse rendering loss, as well as a silhouette consistency term.

Our experimental analysis is divided into two parts. First, we test our normalizing flow-augmented neural implicit shape representation on both synthetic complete and partial depth readings, and demonstrate superior performance with respect to DeepSDF [2]. Second, we test the complete framework on the KITTI [5] benchmark. We set a new state-of-the-art over DSP-SLAM [1] in vision-only mode, and furthermore maintain competitive performance even when lidar information is added. We furthermore demonstrate reasonably good performance while making use of mask constraints, only. We believe that our proposition is of high interest in the ADAS community, which currently gains traction and tendentially restrains itself to the close-to-market alternative of vision-only exteroceptive perception.

II. RELATED WORK

Object-level SLAM: Dame et al. [6] introduce one of the first visual SLAM solutions operating at the level of objects using DCTs of SDFs as a low-dimensional shape representation. Later on, SLAM++ [7] highlights the multi-view pose constraints emanating from object pose estimation and proposes object-level pose graph optimization while relying on a database of shape priors. Zhu et al. [8] again

[1]Motovis Intelligent Technologies, [2]Australian National University, [3]ShanghaiTech University

perform shape optimization within a photometric bundle adjustment objective, and are the first to use a deep neural network as an internal representation (notably for point clouds). Deep-SLAM++ [9] also employs a shape completion network inside SLAM, however performs refinement by discrete selection from multiple forward predictions rather than back-propagating optimization. Node-SLAM [10] is the first object-level SLAM framework making use of DeepSDF [2] within optimization. The latter is also embedded into DSP-SLAM [1]. Liu et al. [11] also make use of implicit neural shape representations, however focus on the prediction of latent codes from multi-sweep lidar scans rather than optimization. More recently, new representations for uncertainty-aware shape representation and optimization were introduced by Liao and Waslander [12], [13], though with a focus on more general shape reconstruction from RGBD rather than vision-only automotive applications. Finally, Han et al. [14] propose the use of neural radiance fields for object-level estimation, and Kong et al. [15] propose to train an MLP for implicit scene representation during SLAM, thereby not imposing priors but merely regularization constraints. DSP-SLAM [1] remains the most related to the present work, and serves as our reference baseline. Note however that the framework optionally uses lidar points, and we compare against both its depth-aware and vision-only variants.

Although the focus of the present work lies on object reconstruction rather than dynamic vehicle perception, it is worth acknowledging the existence of solutions attempting the estimation of dynamic vehicles in the scene by employing trajectory priors [16], [9], [17], [3]. TwistSLAM++ [3] in particular again uses DeepSDF to estimate object shapes from lidar points.

Object shape representations: Although many representations for image or point cloud-based shape prediction have been presented in the literature [18], [19], [20] (e.g. voxel-based, point-cloud based, mesh-based), the biggest advance in terms of differential shape generators that can be embedded as a regularizer into an object-level SLAM framework has been brought along by the introduction of implicit neural field representations. The latter have focussed on scene-level field representations for occupancy [21] or radiance [22], and finally—through the introduction of DeepSDF [2]—individual object shapes. DeepSDF has been the subject of many follow-up contributions aiming at improving the accuracy and compactness of the representation. In a direct follow-up to DeepSDF [2], Chabra et al. [23] employ the representation at scene-level by parametrizing and interpolating SDFs in near-surface voxels, only. In an aim to reduce dissimilarities between generated shapes and a shape collection, Zheng et al. [24] add an end-to-end trainable GAN network to classify shapes as real or fake from both local and global perspectives. In an aim to improve performance under noisy and sparse measurements, Duggal et al. [4] similarly propose the addition of an encoder for robust latent code initialization and a discriminator to induce a learned prior over the predicted SDF. While the addition of a discriminator is helpful, the presently proposed addition of a normalizing flow network forms a more natural architecture to be embedded into back-propagating optimization.

Recently, the introduction of transformers and diffusion models has lead to another wave of powerful differential shape generation networks [25], [26], however moving away from the potentially compact and highly flexible implicit function representations. In an effort to reduce complexity, the community has proposed the tri-planes abstraction [27], [28]. More recently, Yariv et al. propose Mosaic-SDF [29], which—in analogy to Deep Local Shapes [29]—applies the transformer based diffusion model only in local sub-volumes near the surface. While this reduces the size of each tensorial sub-volume to $7\times7\times7$, diffusion-based architectures remain computationally demanding as optimizing a single shape still takes more than 2 minutes on an A100 GPU. As a result, diffusion models are not yet amenable to real-time object-level SLAM.

Normalizing flow: The present work advocates the use of compact, application-specific DeepSDF deviates elegantly augmented by a normalizing flow network to generate reasonable shapes from sparse and noisy visual measurements, and improve convergence behavior of the optimization. Normalizing flow networks have already been promoted in point-cloud oriented representations such as PointFlow [30], DPF-Flow [31], SoftFlow [32], and Go with the flows [33]. Here we make use of iterative *Gaussianization Flow* [34] for latent shape code distribution learning. It uses a repeated composition of trainable kernel layers and orthogonal transformations, can transform any continuous random vector into a Gaussian one, and has demonstrated competitive performance versus several other representations (Real-NVP [35], Glow [36], and FFJORD [37]).

III. Introduction to NF-SLAM

As for many SLAM frameworks proposed over the past years, both DSP-SLAM and the presently proposed NF-SLAM make use of ORB-SLAM2 [38] as a sparse feature-based foundation for efficient tracking and mapping. DSP-SLAM furthermore incorporates lidar depth readings as an optional input, which turns out to be crucial in order to obtain sufficiently accurate object shape estimations. Our newly proposed framework uses only stereo images. However, by averaging temporal information and making use of a normalizing flow-supported implicit representation, we manage to achieve reliable results even in the absence of direct depth readings. In the following, we will go through the notations used in this paper, a complete system overview, as well as a summary of both the front-end and back-end modules.

A. Notations and conventions

The input of the proposed system is given by stereo images. We denote the left and right image of the i-th stereo pair f_{li} and f_{ri}. In order to get semantic information, we utilize an off-the-shelf 2D detector to identify 2D bounding boxes B_{ij} and instance masks M_{ij} for each object in each image. All subsequent operations are conducted based on the left image as the reference. T_o^c refers to the 7-DoF pose

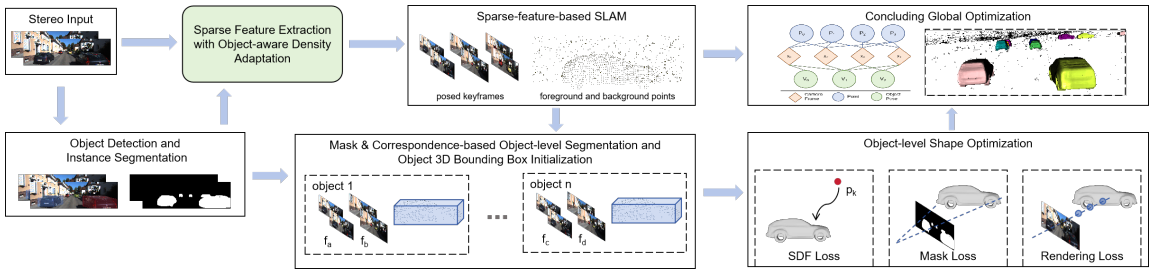


Fig. 1: Overview of our proposed framework for object-level visual SLAM in automotive applications.

of the object. Pixels in the latter are denoted by vectors u. In order to represent shapes, our implicit distance function F_θ makes use of a latent shape code $\mathbf{z} \in \mathbb{R}^{16}$. Let G_ϕ furthermore be the normalizing flow network pre-pended to the MLP. Its input is given by the variable $\mathbf{w} \in \mathbb{R}^{16}$, which ideally obeys a normal distribution. θ and ϕ represent the weights of these two networks, respectively.

B. Framework overview

An overview of the complete framework is illustrated in Figure 1. Each newly incoming stereo frame is first subjected to front-end modules for detecting 2D bounding boxes of objects and their respective instance-level semantic masks [39] as well as sparse ORB keypoints. The latter are then forwarded to ORB-SLAM2 [38] for sparse, stereo visual SLAM. As a result of this module, we will obtain a sparse 3D point cloud and stereo frames all registered within a global reference frame. The segmentation module then checks in which frame and notably mask each of the optimized 3D points is visible, and forms new, instance-level sub-graphs containing adjacent stereo frames that observe the same object and its corresponding subset of 3D points. The extraction of adjacent temporal observations and object-level multi-view graphs is crucial for reliability in the vision-only case as reliance on a single stereo frame may easily lead to suboptimal results.

Once the subgraphs are defined, a combination of 3D point information as well as 2D bounding box coordinates is used in order to initialize a 3D bounding box around each object. Next, we formulate a shape optimization objective based on our normalizing flow-supported implicit neural representation for each individual object. Finally, the reconstructed objects are incorporated into a joint factor graph for global bundle adjustment, aiming to achieve a globally consistent map. Note that, while Figure 1 suggests sequential batch processing of all modules, the modules are indeed all operating incrementally and adding new observations and constraints as new stereo images are being processed.

For the sake of visualization purposes, a fast choice with online capability is given by the marching cubes algorithm.

C. Details on front-end modules

The standard ORB feature extractor from ORB-SLAM2 aims at an as-homogeneous-as-possible feature distribution, which may be benefitial for the robustness and accuracy of the ego-motion estimation. However, given that object observations often make up for only a small part of the image, it naturally limits the number of sparse keypoints that we extract on object surfaces, thereby limiting the potential given by some of the employed loss functions. We therefore implement a modified feature extractor that revisits 2D Regions Of Interest (ROIs) corresponding to object bounding boxes, and significantly enriches the feature extraction in those regions by applying a vastly reduced threshold. As a result, we naturally obtain more correspondences and optimized 3D points on object surfaces. This in turn provides the shape optimizer with an increased number of constraints, and as a result leads to better overall shape estimation results. Note that this operation is performed in both the left and the right image, which also explains the need to extract semantic information in both views.

A second module of interest in the front-end is given by the object pose initialization module. We make use of the 3D bounding box initialization method proposed by Li et al. [16], which is taylored to stereo cameras. However, the method depends on a view-point classifier, which we replace by a discretization of the orientation coming out of a principle component analysis of the object's 3D point cloud.

D. Details on back-end modules

The back-end module consists of two sub modules. The first one takes sets of adjacent frames observing a single object along with the relative object frame definition, the 3D object points, corresponding image keypoints, the masks, and the bounding boxes, and feeds them to a per-object shape optimization module. The objective of the shape optimization is composed of up to three losses, meaning an SDF loss (i.e. sparse 3D point-to-surface distances), a silhouette inconsistency loss (i.e. discrepancies between rendering-based occupancies and measured occupancies), and a rendering loss (i.e. differences between rendered depths and optimized depths). Note that all three losses are formulated as a function of our normalizing-flow supported, implicit neural shape representation. The shape optimization method forms the core of our contribution, and more details on representation and optimization are provided in Section IV.

The concluding stage is given by global optimization of a joint factor graph that includes point features, camera poses, and object poses. The original graph optimization module from ORB-SLAM2 already performs bundle adjustment and loop closure to ensure a globally consistent map. As new objects are being detected, they are incorporated as further nodes in a joint factor graph, and the edges are formed by the corresponding relative object pose estimates. The main

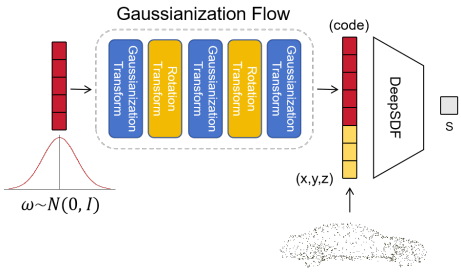


Fig. 2: Implicit neural shape representation used in the present work. The architecture is composed of a DeepSDF decoder preceded by a normalizing flow network. Given an input latent code $\mathbf{w}$ and a sampling point $\{x, y, z\}$, the network generates the 3D Euclidean distance between the sampled point and the object surface.

target of this extension is to improve the accuracy of the relative location of each object reference frame.

IV. Vehicle Shape Optimization with Normalizing-Flow Supported Implicit Fields

Our proposed vehicle shape optimization makes use of an implicit neural shape generator. The present section introduces the architecture of the network, the constraints employed during optimization, and the concluding optimization itself.

A. Network Architecture

Our shape optimizer internally incorporates two types of networks: normalizing flow and DeepSDF. It aims to optimize latent vectors from a standard Gaussian distribution into a high-dimensional representation that best aligns with the current point cloud shape. For each queried point, it computes its signed distance function value.

The normalizing flow module G_ϕ is designed to generate shape latent vectors from variables that adhere to a standard Gaussian distribution. Drawing inspiration from the work of Meng et al. [34], we use Gaussianization flow as our normalizing flow embedding. Gaussianization flow is a trainable extension of Rotation-Based Iterative Gaussianization. It is constructed by stacking trainable kernel layers and orthogonal matrix layers alternatively, as Fig.2 shows.

We employ DeepSDF as our shape decoder F_θ, which can estimate the SDF value for each point from a latent vector $\mathbf{z} \in \mathbb{R}^{16}$. The normalized code $\mathbf{w}$ hence also has only 16 dimensions. Our DeepSDF derivate has only 8 fully connected layers, and the normalizing flow module employs three Gaussianization transform layers with interleaving rotation transform layers. The network is trained in two stages. We first train the DeepSDF network, and subsequently train the normalizing flow network.

B. Optimization Constraints

During shape reconstruction, we fix the network weights θ, ϕ. For each instance, assuming $\mathbf{z}$ represents the latent shape code and $p_\mathbf{z}$ its data distribution, the normalizing flow module executes a bijective, differentiable transformation of $\mathbf{z}$ into $\mathbf{w} = G_\phi^{-1}(\mathbf{z})$. We first initialize $\mathbf{w}$ with zero-value vector from scratch. Subsequently, the latent code $\mathbf{z}$ can be generated using the normalizing flow network. Next, each test point is concatenated with $\mathbf{z}$, and jointly fed into DeepSDF. The trained DeepSDF is capable of predicting the SDF value for each test point $\mathbf{p}$, i.e.

$$s = F_\theta(T_c^o \mathbf{p}, G_\phi(\mathbf{w})) \tag{1}$$

Under the supervision of certain loss functions, we then try to find an optimal $\hat{\mathbf{w}}$ and $\hat{T}_o^c$. With the optimal shape code in hand, we can apply the marching cubes algorithm to get a full 3D mesh.

We utilize the following constraints for shape optimization.

1) SDF surface constraint: Each 3D point $\mathbf{p}_k$ corresponding to a detected sparse keypoint can be transformed into its corresponding canonical object reference frame using $P_k^o = T_c^o \mathbf{p}_k$. In an ideal scenario, the SDF value of points on the object surface is expected to be 0, so we can formulate the surface loss by

$$L_{surf} = \frac{1}{N} \sum_{k=1}^{N} \| F_\theta(T_c^o \mathbf{p}_k, G_\phi(\mathbf{w})) \|^2. \tag{2}$$

2) Foreground mask constraint: Inspired by work of Tulsiani et al. [40], we formulate a differentiable loss function termed as 'view consistency', which quantifies the disparity between the reconstructed 3D shape and its corresponding observation in the image. During the ray tracing process, we use e_i to describe the probability that the i^{th} sampling point is empty, where 1 means being empty and vice visa. According to Equation 3, it can be computed from the SDF value s_i by

$$e_i = \begin{cases} 0 & s_i < -\sigma \\ \frac{1}{2} + \frac{s_i}{2\sigma} & |s_i| \le \sigma \\ 1 & s_i > \sigma \end{cases}. \tag{3}$$

Suppose the ray r passes through N_r sampling points. The ray may hit one of the N_r sampling points or just escape. We use the variable t_r to represent the sampling point at which the ray probabilistically terminates, where $t_r = N_r + 1$ means the ray doesn't hit the surface

$$p(t_r = i) = \begin{cases} (1 - e_i) \prod_{j=1}^{i-1} e_j & i \le N_r \\ \prod_{j=1}^{N_r} e_j & i = N_r + 1 \end{cases} \tag{4}$$

To combine the known information from the object mask for each ray, we use $m_r \in \{0, 1\}$, where $m_r = 0$ means the ray r intersects corresponding to a pixel within the object mask and thus is supposed to hit the surface, and $m_r = 1$ means the ray doesn't hit the target object. We can finally assign a cost $\psi_r^{mask}(i)$ to the event $(t_r = i)$, thereby punishing predictions inconsistent with observations. We have

$$\psi_r^{mask}(i) = \begin{cases} m_r & i \le N_r \\ 1 - m_r & i = N_r + 1 \end{cases}. \tag{5}$$

Finally, our mask constraint is given by,

$$L_{mask} = \frac{1}{|\Omega_{MB}|} \sum_{r \in \Omega_{MB}} \sum_{i=1}^{N_r+1} p(t_r = i) \psi_r^{mask}(i) \quad , \qquad (6)$$

where $\Omega_{MB} = \Omega_M \cup \Omega_B$ means the collection of pixels located within the mask region and those outside the Mask region but inside the 2D bounding box.

3) Depth constraint: After estimating the depth d_u of points through stereo triangulation or bundle adjustment, we can formulate an additional loss that considers the rendered depth at such points. Note that such rendering loss is not to be confused with the surface loss, as the rendering loss implicitly forces observed depth points to align with the unoccluded part of the surface, while the surface loss simply chooses the distance to the nearest point anywhere on the surface. During ray tracing along each ray, each sampled depth is $d_i = d_{min} + (i-1)\Delta d$, and the sampling interval step size is typically denoted by $\Delta d = (d_{max} - d_{min})/(N_r - 1)$. By combining Equation 4, we can compute the rendered depth for each pixel. The rendering depth loss is finally given by

$$L_{depth} = \frac{1}{|\Omega_{SB}|} \sum_{u \in \Omega_{SB}} (d_u - \sum_{i=1}^{N_r+1} p_i d_i)^2 \quad , \qquad (7)$$

where $\Omega_{SB} = \Omega_S \cup \Omega_B$ consists of points with depth lying on the object surface, as well as points not on the surface but still within the 2D bounding box.

C. Optimization Process

Due to the inaccuracies in single-frame image observations, we devise a joint multi-frame optimization strategy to reconstruct the shape. The points utilized in the surface constraints are derived from the fusion of 3D points across Q frames. We compute the mask loss for each frame, and average them to obtain the final mask term for the object. A similar strategy is applied for the depth term. The last term is the regularization term.

$$L = \lambda_1 L_{surf} + \sum_{i=1}^{Q} \left(\lambda_2 L_{mask} + \lambda_3 L_{depth} \right) + \lambda_4 \|\mathbf{w}\|_2^2 \qquad (8)$$

Given the quadratic nature of all terms, we apply a Gauss-Newton optimization strategy employing analytical Jacobians.

V. Experiments

Our experiments split up into two parts. We first test our newly proposed implicit neural shape representation and compare it against similarly dimensioned variants of Deep SDF. Second, we test the entire object-level SLAM framework on KITTI sequences and compare results against DSP-SLAM.

A. Shape generator evaluation on synthetic datasets

To assess whether the reconstructed shape closely matches the original point cloud, we conduct experiments on the synthesized dataset ShapeNet [41]. We retrained the DeepSDF network with a 16 dimensional latent code and compare our new architecture against this variant to solely assess the impact of adding the normalizing flow network. Additionally, for both networks, we evaluate the impact of different optimization methods during inference, including Gauss-Newton and the Pytorch Adam optimizer.

TABLE I: Quantitative results obtained on complete point clouds.

Method\ Metric	Chamfer Distance		
	Median	Mean	Std
Our GN	0.2023	**0.2588**	**0.2785**
Our Adam	0.2078	0.2905	0.3231
DeepSDF GN	**0.1803**	0.2921	0.3339
DeepSDF	0.2201	0.3072	0.3343

TABLE II: Quantitative results obtained on partial point clouds.

Method\ Metric	Chamfer Distance		
	Median	Mean	Std
Our GN	0.4017	**0.4470**	0.2017
Our Adam	0.4128	0.4693	**0.1880**
DeepSDF GN	**0.3982**	0.4760	0.2301
DeepSDF	0.4809	0.5169	0.2107

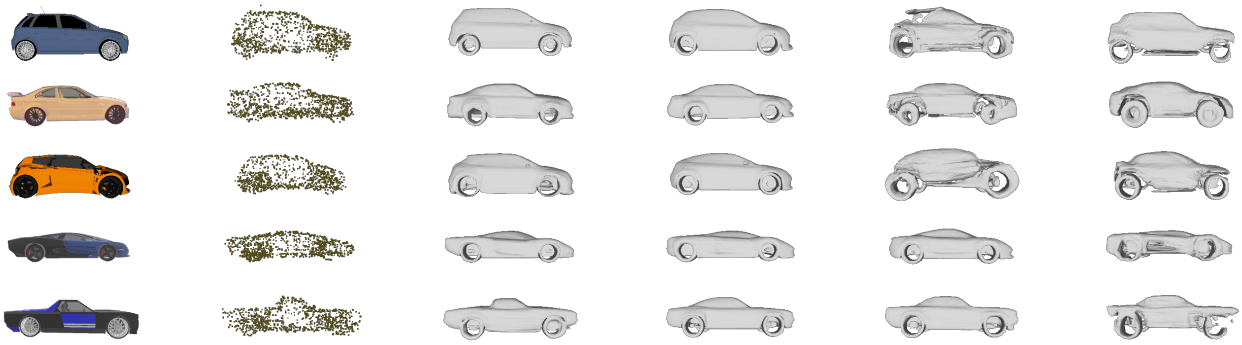
1) Test on ShapeNet with Complete Point Clouds: For each analyzed object from ShapeNet, we extract surface points following the data preparation steps outlined in the original DeepSDF paper. We refer to this as the "complete point cloud", as it is an evenly distributed set of points lying on the surface of the object. The optimization simply employs the priorly mentioned surface distance evaluated for all points in the point cloud.

We use the Chamfer distance as an evaluation metric. We evaluate the bidirectional chamfer distance between the points sampled on the zero-level set of the optimized shape (set S_1) and the original surface points extracted from ShapeNet data (set S_2), i.e.

$$d_{CD}(S_1, S_2) = \frac{1}{|S_1|} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \frac{1}{|S_2|} \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2 \qquad (9)$$

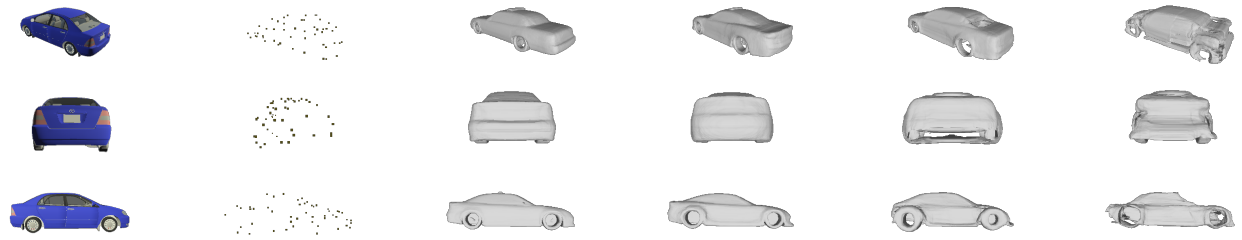
We use 1000 surface points for shape reconstruction. Table I lists the results obtained for complete point clouds, note that for statistical purposes, all values have been multiplied by 1000. and Figure 3 shows qualitative examples. As can be observed, our method consistently produces reasonable shapes and delivers the best results in terms of the mean and standard deviation, thus indicating better robustness and convergence ability.

2) Tests on ShapeNet with Partial Point Clouds: In order to simulate real-world-like scenarios where only a portion of the point cloud is visible, we render partial point clouds from 10 different views and repeat the surface distance based optimization for all networks and optimization variants over these sample sets individually. Even though we only use partial point clouds to perform the shape optimization, we continue to evaluate the same bi-directional chamfer distance to evaluate the complete generated shape. For each object, we notably compute the average chamfer distance across these



(a) GT obj (b) GT Points (c) Our GN (d) Our Adam (e) DeepSDF GN (f) DeepSDF Adam

Fig. 3: Shape optimization results for complete point clouds taken from ShapeNet. From left to right: Original model, point cloud samples, our proposed generator with normalizing flow optimized with Gauss-Newton, the same generator optimized with Adam, DeepSDF results using Gauss-Newton, and DeepSDF results using Adam optimizer.



(a) GT obj (b) GT Points (c) Our GN (d) Our Adam (e) DeepSDF GN (f) DeepSDF Adam

Fig. 4: Shape optimization results for partial point clouds taken from ShapeNet. From left to right: Original model, point cloud samples, our proposed generator with normalizing flow optimized with Gauss-Newton, the same generator optimized with Adam, DeepSDF results using Gauss-Newton, and DeepSDF results using Adam optimizer.

10 perspectives and use it as the final result for that object. Table II again lists the median, mean, and standard deviation for all objects. As can be observed, we continue to exhibit low median errors and strong robustness as indicated by significantly lower mean and standard deviations. As further supported by the qualitative results in Figure 4, the addition of normalizing flow provokes the consistent generation of more reasonable shapes, in the case of partial point cloud samples. Note that, in this experiment, only 50 surface points are used for reconstruction.

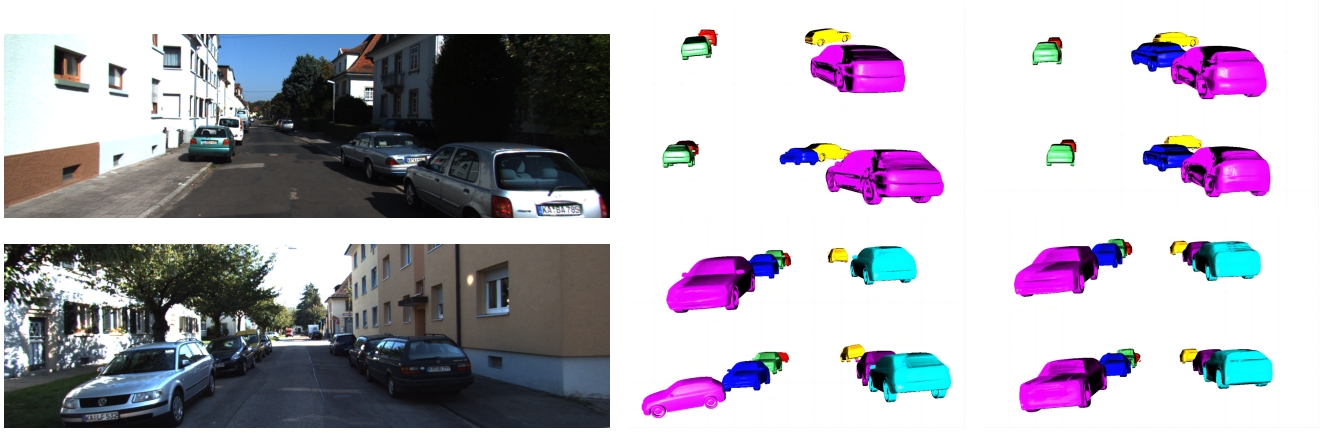
B. Test on real world dataset

To evaluate the full system, we do experiments on the KITTI odometry dataset. As our focus lies on the quality of the vehicle reconstruction, we assess results from shape completeness and accuracy. Shape completeness analyses whether or not the reconstructed shape represents a complete, reasonable vehicle rather than just wheels or a partitioned mesh with artefacts. The accuracy measures the proximity of the reconstructed part and the original point cloud. We test 10 sequences which contain a significant number of vehicles. We compare four methods: DSP-SLAM with lidar as input, DSP-SLAM* with only stereo images, NF-SLAM, and NF-SLAM* with only mask constraint. Note that in DSP-SLAM, a 64-dimensional latent vector is utilized as the shape representation, whereas in our approach, we employ a 16-dimensional latent vector.

1) Completeness Evaluation: In order to assess the completeness of the object shapes, we use the 3D IoU (Intersection over Union) which can provide a quantitative metric to assess the size and thus completeness of the reconstructed object shapes. For each reconstructed vehicle, we estimated its 3D bounding box and calculate the 3D IoU with the 3D bounding box detected from the lidar point cloud. For each sequence, we computed the median, mean, and standard deviation. Quantitative results for all 10 sequences are summarized in Table III.

2) Accuracy Evaluation: To conclude, the lidar point clouds— captured by a Velodyne laser scanner and utilized to create the ground-truth data in the KITTI dataset—can be used for a more detailed assessment of the accuracy of the reconstructed part of the shape for each vehicle. To accomplish this, we transform the lidar point cloud into individual object coordinate frames and then segment out the subset of the points that resides within the vehicle's 3D reference bounding box.

Owing to the nature of real-world scenarios, only a partial view of the car is visible at any given moment. This implies



(a) KITTI images (b) DSP-SLAM / DSP-SLAM* (c) NF-SLAM / NF-SLAM*

Fig. 5: Qualitative results obtained on KITTI dataset. In subfigure (b), DSP-SLAM is positioned above, DSP-SLAM* is positioned below. In subfigure (c), NF-SLAM is positioned above, NF-SLAM* is positioned below.

TABLE III: Assessment of object completeness using the 3D IoU between the inferred and ground truth bounding boxes.

	DSP-SLAM			DSP-SLAM*			NF-SLAM			NF-SLAM*		
Seq	**Median**	**Mean**	**Std**	**Median**	**Mean**	**Std**	**Median**	**Mean**	**Std**	**Median**	**Mean**	**Std**
00	0.8059	0.8004	**0.0528**	0.7134	0.6265	0.2064	**0.8402**	**0.8284**	0.0733	0.8092	0.7810	0.1257
05	0.7855	0.7855	0.0619	0.7656	0.7344	0.1559	**0.8426**	**0.8350**	**0.0528**	0.8034	0.7925	0.0992
06	0.7943	**0.7960**	**0.0525**	0.7682	0.7161	0.1391	**0.8078**	0.7816	0.1140	0.7876	0.7763	0.0989
07	0.8058	0.7902	0.0845	0.7341	0.6670	0.1992	**0.8115**	**0.8087**	**0.0647**	0.8070	0.7855	0.1150
08	0.8142	0.8050	0.0596	0.7667	0.6525	0.2572	**0.8394**	**0.8326**	**0.0574**	0.8228	0.8040	0.1099
10	0.8018	**0.7931**	**0.0659**	0.7586	0.6542	0.2295	**0.8241**	0.7888	0.1373	0.8037	0.7436	0.1749
11	0.7817	0.7813	**0.0416**	0.7768	0.7006	0.2071	**0.8440**	**0.8425**	0.0607	0.7966	0.7943	0.0829
15	0.8097	0.8039	0.0570	0.7349	0.6526	0.2220	0.8189	**0.8316**	**0.0512**	**0.8325**	0.8038	0.1168
16	0.8090	0.8024	**0.0477**	0.7802	0.7267	0.1427	**0.8471**	**0.8289**	0.0812	0.8060	0.7803	0.1372
19	0.8080	0.7965	0.0818	0.7444	0.6775	0.1799	**0.8319**	**0.8289**	**0.0617**	0.8099	0.7884	0.1078
Mean	0.8015	0.7954	**0.0605**	0.7543	0.6808	0.1939	**0.8307**	**0.8207**	0.0754	0.8078	0.7849	0.1168

TABLE IV: Unidirectional Chamfer Distances evaluating the accuracy of the reconstructed shapes on KITTI.

	DSP-SLAM			DSP-SLAM*			NF-SLAM			NF-SLAM*		
Seq	**Median**	**Mean**	**Std**	**Median**	**Mean**	**Std**	**Median**	**Mean**	**Std**	**Median**	**Mean**	**Std**
00	0.2517	0.3266	**0.4428**	1.1405	3.1542	5.3138	**0.2171**	**0.3218**	0.5123	0.2657	0.4961	0.8832
05	**0.2658**	0.3449	0.2268	0.3386	0.7271	1.0228	0.3013	**0.3183**	**0.1700**	0.3303	0.5548	0.8193
06	**0.2225**	**0.2660**	**0.1652**	0.5065	2.3795	8.5699	0.2348	1.1300	5.9138	0.3089	0.4560	0.5099
07	**0.2779**	**0.3243**	**0.2192**	0.6718	2.0225	5.8163	0.2985	0.3838	0.3329	0.3620	0.4558	0.4943
08	0.3040	0.3008	**0.1057**	0.4501	1.5392	2.1141	**0.2235**	**0.2698**	0.1619	0.2871	0.3831	0.3104
10	0.3365	**0.4190**	**0.3355**	0.6131	2.9308	7.9568	**0.2211**	0.7863	1.9294	0.3111	1.7130	4.8879
11	**0.2479**	**0.3066**	0.1751	0.4041	1.0182	1.5594	0.3009	0.3085	**0.1650**	0.3174	0.3391	0.1792
15	0.3049	0.3855	0.3959	0.4148	1.4261	2.1337	**0.2628**	**0.2819**	**0.1121**	0.2802	0.9101	3.7743
16	0.2333	**0.2688**	0.2224	0.3421	1.0811	1.9455	**0.2329**	0.2781	**0.1673**	0.2575	0.4779	0.5647
19	0.2341	0.2549	**0.1349**	0.5741	2.3888	5.5756	**0.2158**	**0.2467**	0.1522	0.2494	0.3449	0.3482
Mean	0.2678	**0.3197**	**0.2423**	0.5455	1.8667	4.2007	**0.2508**	0.4325	0.9616	0.2969	0.6130	1.2771

that our observations are limited to partial point samples. Hence, we opted for the unidirectional chamfer distance as our evaluation criterion and find the distance for each of our input samples to the ground truth vehicle shape, i.e.

$$d_{UCD}(S_1, S_2) = \frac{1}{|S_1|} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 \tag{10}$$

Quantitative results are summarized in Table IV. For the sake of convenience, the numerical values in the table have been multiplied by 100. Qualitative results are shown in Figure 5. From the experimental results, it can be observed that when the input is limited to only stereo images, NF-SLAM is comparable to DSP-SLAM using lidar data in most cases. NF-SLAM* achieves decent results using only mask loss, whereas the performance of the DSP-SLAM* significantly decreases.

VI. CONCLUSION

We propose a novel implicit shape generation model making use of normalizing flow for improved robustness, convergence and overall performance in situations of limited input samples. Combined with a multi-frame optimization scheme, the module achieves reasonable vehicle shape reconstructions on par with lidar-based optimization schemes. We believe the proposed architecture to be of potentially

broad interest in vision-only object-level perception tasks, both with and without a focus on automotive applications.

ACKNOWLEDGMENT

We would like to acknowledge the funding support provided by project 62250610225 by the Natural Science Foundation of China, as well as projects 22DZ1201900, 22ZR1441300, and dfycbj-1 by the Natural Science Foundation of Shanghai.

REFERENCES

[1] J. Wang, M. Rünz, and L. Agapito, "DSP-SLAM: Object oriented SLAM with deep shape priors," in *Proceedings of the International Conference on 3D Computer Vision (3DV)*, 2021.

[2] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 165–174.

[3] M. Gonzalez, E. Marchand, A. Kacete, and J. Royan, "Twistslam++: Fusing multiple modalities for accurate dynamic semantic slam," 2022.

[4] S. Duggal, Z. Wang, W.-C. Ma, S. Manivasagam, J. Liang, S. Wang, and R. Urtasun, "Mending neural implicit modeling for 3d vehicle reconstruction in the wild," 2022, pp. 1900–1909.

[5] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[6] A. Dame, V. Prisacariu, C. Ren, and I. Reid, "Dense reconstruction using 3d object shape priors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[7] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[8] R. Zhu, C. Wang, C.-H. Lin, Z. Wang, and S. Lucey, "Semantic photometric bundle adjustment on natural sequences," 2017.

[9] L. Hu, W. Xu, H. Shi, K. Huang, and L. Kneip, "Deep-SLAM++: Object-level RGBD SLAM based on class-specific deep shape priors," 2019.

[10] E. Sucar, K. Wada, and A. Davison, "NodeSLAM: Neural object descriptors for multi-view shape reconstruction," in *Proceedings of the International Conference on 3D Computer Vision (3DV)*, 2020.

[11] Y. Liu, G. Zhu, K. Wu, Y. Ren, B. Liu, Y. Liu, and J. Shan, "MV-DeepSDF: Implicit Modeling with Multi-Sweep Point Clouds for 3D Vehicle Reconstruction in Autonomous Driving," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023, pp. 8306–8316.

[12] Z. Liao, J. Yang, J. Qian, A. P. Schoellig, and S. L. Waslander, "Uncertainty-aware 3d object-level mapping with deep shape priors," 2023.

[13] *Multi-view 3D Object Reconstruction and Uncertainty Modelling with Neural Shape Prior*, 2024.

[14] X. Han, H. Liu, Y. Ding, and L. Yang, "RO-MAP: Real-Time Multi-Object Mapping With Neural Radiance Fields," *IEEE Robotics and Automation Letters*, vol. 8, no. 9, pp. 5950–5957, 2023.

[15] X. Kong, S. Liu, M. Taher, and A. J. Davison, "vMAP: Vectorised Object Mapping for Neural Field SLAM," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[16] P. Li, T. Qin, and S. Shen, "Stereo vision-based semantic 3d object ego-motion tracking for autonomous driving," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[17] B. Bescos, C. Campos, J. D. Tardos, and J. Neira, "DynaSLAM II: Tightly-Coupled Multi-Object Tracking and SLAM," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5191–5198, 2021.

[18] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] Y. Liao, S. Donne, and A. Geiger, "Deep marching cubes: Learning explicit surface representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] T. Groueix, M. Fisher, V. Kim, B. Russell, and M. Aubry, "Atlasnet: A papier-mache approach to learning 3d surface generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[21] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[22] B. Mildenhall, P.-P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[23] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe, "Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[24] X.-Y. Zheng, Y. Liu, P.-S. Wang, and X. Tong, "SDF-stylegan: Implicit sdf-based stylegan for 3d shape generation," in *In Comput. Graph. Forum (SGP)*, 2022.

[25] K.-H. Hui, R. Li, J. Hu, and C.-W. Fu, "Neural wavelet-domain diffusion for 3d shape generation," in *In Proceedings of SIGGRAPH Asia*, 2022.

[26] N. Müller, Y. Siddiqui, L. Porzi, S. Rota Bulo, P. Kontschieder, and M. Nie, "Diffrf: Rendering-guided 3d radiance field diffusion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[27] J. R. Shue, E. R. Chan, R. Po, Z. Ankner, and J. Wu, "3D neural field generation using triplane diffusion," 2022.

[28] T. Wang, B. Zhang, T. Zhang, S. Gu, J. Bao, T. Baltrusaitis, J. Shen, D. Chen, F. Wen, Q. Qifeng Chen, and B. Guo, "Rodin: A generative model for sculpting 3d digital avatars using diffusion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[29] L. Yariv, O. Puny, N. Neverova, O. Gafni, and Y. Lipman, "Mosaic-SDF for 3D Generative Models," 2023.

[30] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 4541–4550.

[31] R. Klokov, E. Boyer, and J. Verbeek, "Discrete point flow networks for efficient point cloud generation," in *European Conference on Computer Vision*. Springer, 2020, pp. 694–710.

[32] H. Kim, H. Lee, W. H. Kang, J. Y. Lee, and N. S. Kim, "Softflow: Probabilistic framework for normalizing flow on manifolds," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 388–16 397, 2020.

[33] J. Postels, M. Liu, R. Spezialetti, L. Van Gool, and F. Tombari, "Go with the flows: Mixtures of normalizing flows for point cloud generation and reconstruction," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 1249–1258.

[34] C. Meng, Y. Song, J. Song, and S. Ermon, "Gaussianization flows," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 4336–4345.

[35] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[36] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," *Advances in neural information processing systems*, vol. 31, 2018.

[37] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," *arXiv preprint arXiv:1810.01367*, 2018.

[38] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics (T-RO)*, vol. 33, no. 5, pp. 1255–1262, 2017.

[39] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[40] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik, "Multi-view supervision for single-view reconstruction via differentiable ray consistency," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2626–2634.

[41] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.