LiSu: A Dataset and Method for LiDAR Surface Normal Estimation

Dušan Malić[1,2]   Christian Fruhwirth-Reisinger[1,2]   Samuel Schulter[3,†]   Horst Possegger[1,2]

[1]Christian Doppler Laboratory for Embedded Machine Learning
[2]Institute of Visual Computing, Graz University of Technology
[3]Amazon

{dusan.malic, reisinger, possegger}@tugraz.at

(a) LiSu dataset sample   (b) Our method applied on a Waymo frame   (c) SHS-Net [31] applied on a Waymo frame
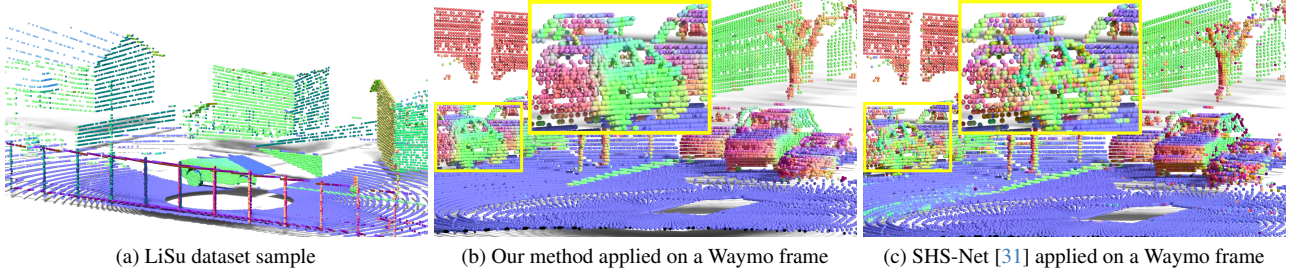
Figure 1. Our synthetic LiSu dataset (a) enables focusing research on the challenging task of LiDAR surface normal estimation. When combined with our proposed method, we achieve state-of-the-art results (b) on challenging real-world datasets like Waymo Open Dataset [48], outperforming the current state-of-the-art SHS-Net [31] (c). Best viewed in color on screen.

Abstract

While surface normals are widely used to analyse 3D scene geometry, surface normal estimation from LiDAR point clouds remains severely underexplored. This is caused by the lack of large-scale annotated datasets on the one hand, and lack of methods that can robustly handle the sparse and often noisy LiDAR data in a reasonable time on the other hand. We address these limitations using a traffic simulation engine and present LiSu, the first large-scale, synthetic Li-DAR point cloud dataset with ground truth surface normal annotations, eliminating the need for tedious manual labeling. Additionally, we propose a novel method that exploits the spatiotemporal characteristics of autonomous driving data to enhance surface normal estimation accuracy. By incorporating two regularization terms, we enforce spatial consistency among neighboring points and temporal smoothness across consecutive LiDAR frames. These regularizers are particularly effective in self-training settings, where they mitigate the impact of noisy pseudo-labels, enabling robust real-world deployment. We demonstrate the effectiveness of our method on LiSu, achieving state-of-the-art performance in LiDAR surface normal estimation. Moreover, we showcase its full potential in addressing the challenging task of synthetic-to-real domain adaptation, leading to improved neural surface reconstruction on real-world data.

1. Introduction

Representing 3D surfaces by their surface normals is beneficial for a wide range of computer vision tasks, such as neural rendering [51, 56, 63], robotics [3, 40, 46, 65], autonomous driving [15, 38, 50], *etc*. While significant strides have been made in monocular surface normal estimation, particularly with large models trained on extensive datasets [2, 24, 26], research on surface normal estimation from LiDAR point clouds in the autonomous driving domain remains limited.

Existing methods for point cloud surface normal estimation [14, 32, 33, 71] are tailored for small-scale CAD datasets, such as PCPNet [21] dataset, and often based on non-scalable architectures like PointNet [42]. Consequently, these methods struggle with large-scale point clouds and require either down-sampling or partitioning, which results in increased runtime. They often assume dense and uniform point distributions, making them ill-suited for LiDAR data, which is characterized by sparsity, non-uniformity, and noise. Moreover, the evaluation of these methods on LiDAR data is hindered by the lack of publicly available datasets with surface normal annotations.

To overcome these limitations, we present LiSu, a novel synthetic LiDAR dataset targeted for research on surface normal estimation. We leverage CARLA [13], a versatile simulation environment offering diverse urban and rural landscapes, including downtown areas, small towns, and multi-lane highways. By extending CARLA's LiDAR sensor to capture not only point locations but also surface normal vectors, we curate an extensive dataset of roughly 50k frames.

Leveraging the strong feature extraction capabilities of

Point Transformer V3 (PTv3) [58], we propose a single-step approach for surface normal estimation. Our novel method explicitly accounts for the inherent noise in LiDAR data by incorporating two regularization terms: spatial consistency and temporal consistency. These terms enforce smoothness and temporal coherence, respectively, leading to more robust and accurate normal estimates across consecutive frames. Importantly, our method can be straightforwardly applied to domain adaptation scenarios, where our regularization terms effectively alleviate the adverse effects of noisy pseudo-labels in self-supervised learning [61, 64, 66, 68, 69]. This enables us to successfully bridge the synthetic-to-real domain gap, making our model suitable for real-world deployment.

We evaluate our method on our LiSu dataset, benchmarking it against two classical and four state-of-the-art methods for point cloud surface normal estimation. In the absence of real-world datasets with surface normal annotations, we assess the effectiveness of our adaptation method by applying it to the real-world downstream task of neural surface reconstruction [22, 52, 54, 63]. For this, we integrate our surface normal estimator model as an oracle into a LiDAR-only surface reconstruction method [66], demonstrating a substantial performance improvement.

In summary, our main contributions are:
- Extending CARLA's LiDAR sensor to capture surface normals, enabling the generation of synthetic datasets for LiDAR-based surface normal estimation methods.
- LiSu: A novel, synthetic LiDAR dataset for autonomous driving, uniquely featuring labeled surface normals.
- Method employing data term regularizers to explicitly model spatiotemporal dependencies within autonomous driving datasets
- Extensive evaluation on both LiSu and real-world data.
- Publicly accessible code, dataset, and trained models at https://github.com/malicd/LiSu

2. Related Work

Surface normal estimation from 3D point clouds involves the computation of the normal vector at each point on a 3D surface. Traditional methods achieve this by fitting a geometric primitive, such as a tangent plane [23], jets [7] or sphere [20], to a local neighborhood of the query point. The normal vector of this fitted primitive is then used as an estimate of the surface normal at the query point. More recently, approaches leverage deep learning models to enhance surface fitting [4, 14, 27, 28, 72]. Building upon the PointNet architecture [42], several methods [5, 21, 29] directly regress the surface normal vector from input point clouds.

These methods commonly utilize the PCPNet dataset [21], which contains 30 synthetic objects. Each object consists of 100k points representing geometric shapes and figurines. To mitigate the limited data, Guerrero *et al*. [21] introduced a local patch sampling strategy, where N-point patches are extracted from the objects and used for training and inference. However, this approach suffers from two limitations: patches from the same object share an underlying distribution, and complete shape estimation necessitates multiple inference steps. The lack of data diversity and iterative inference routine render such approaches unsuitalbe for large-scale LiDAR point clouds.

Surface normal estimation from LiDAR point clouds remains a challenging problem due to the data's inherent sparsity, non-uniformity, and noise. Traditional surface fitting methods often struggle with these characteristics. Badino *et al*. [1] addressed this by extending plane fitting to LiDAR range images and incorporating a normalization module for noise reduction. More recent approaches, such as Bogoslavskyi *et al*. [6], leverage LiDAR scan lines to define local neighborhoods and infer surface normals.

With the rise of autonomous driving datasets (*e.g*. [16, 48]), data from calibrated LiDAR and camera have become widely accessible. This has spurred the development of multimodal methods like Lin *et al*. [34] and DeepLiDAR [43], which leverage both modalities. Scheuble *et al*. [45] introduced PolLidar, a novel LiDAR sensor capable of capturing time-resolved polarimetric wavefronts. They demostrate that surface normals can be derived from LiDAR characteristics beyond traditional geometric cues. However, these methods rely on calibrated multi-sensor setups or specialized hardware, limiting their adoption in real-world applications.

The field of LiDAR surface normal estimation is hindered by the lack of a standardized benchmark dataset. While the aforementioned works have made notable strides in addressing this, their datasets still exhibit limitations. For example, DeepLiDAR primarily focuses on image-based tasks and employs LiDAR merely as sparse depth information, not providing LiDAR normals. Additionally, the datasets of PolLidar [45] and Lin *et al*. [34] are relatively small (1969 and 151 frames, respectively) and may not fully capture the diversity of real-world scenarios. Crucially, these datasets were *not publicly available* at the time of this submission.

To address these challenges, we introduce LiSu, a synthetic LiDAR dataset of 50k frames with surface normal annotations, freely available to our research community. Additionally, we propose a novel method to effectively leverage synthetic and unlabeled real-world data, enabling the development of robust models that generalize well to real-world scenarios.

Synthetic-to-Real Unsupervised Domain Adaptation (UDA) for LiDAR point clouds transfers knowledge from models trained on synthetic source data to real-world target datasets [60, 64, 66, 67, 70]. Self-training is a widely employed UDA technique for various LiDAR-based tasks [8, 39, 44, 61, 66, 68, 69]. Our method closely aligns with existing UDA techniques. However, we introduce a novel spatial and temporal regularization strategy to explicitly address the

Dataset	# Beams	PPF	# Fr.	SN	Pub.
Waymo [49]	64	160k	230k	✗	✓
PolLidar [45]	150	-	1.9k	✓	✗
Lin *et al.* [34]	-	-	151	✓	✗
LiSu	64	100k	50k	✓	✓

Table 1. Overview of autonomous driving datasets, including Li-DAR sensor configurations (number of beams, points per frame (PPF)), total frames (Fr.), surface normal (SN) annotation availability, and public (Pub.) accessibility.

inherent noise in pseudo-labels.

Neural implicit surface reconstruction parametrizes 3D surfaces as Signed Distance Functions (SDFs) predicted by a multi-layer perceptron (MLP) [41, 47, 54]. A common optimization strategy involves guiding the learning objective with surface normal cues estimated with monocular off-the-shelf networks [19, 22, 52, 63]. These approaches leverage the fact that SDF gradients at surface points correspond to inward surface normals [25]. Naturally, the quality of normal estimates plays a crucial role [19]. While LiDAR-only methods (*e.g.* [66]) may neglect surface normal regularization, our work underscores the substantial benefits of precise surface normal estimations for superior surface reconstruction.

3. Surface Normals from LiDAR Point Clouds

To enable LiDAR-based surface normal estimation research, we introduce LiSu (Sec. 3.1), a large-scale synthetic Li-DAR dataset with surface normal annotations. Moreover, we propose a novel method (Sec. 3.2) which leverages the spatiotemporal nature of autonomous driving data through two data term regularizers, achieving state-of-the-art performance on LiDAR surface normal estimation task. This method also serves as a robust signal for mitigating pseudo-label noise in self-supervised learning.

3.1. LiSu: LiDAR Surface Normal Dataset

We generate our dataset using CARLA [13], a simulation framework based on the Unreal Engine. Specifically, we leverage nine of CARLA's twelve pre-built maps, excluding two reserved for the *CARLA Autonomous Driving challenges* and one undecorated map with low geometric detail (*i.e.* without buildings, sidewalks, *etc.*). These selected maps represent diverse urban and rural environments, including downtown areas, small towns, and multi-lane highways. For each simulation, we populated the scenes with a large number of dynamic actors, such as vehicles (cars, trucks, buses, vans, motorcycles, bicycles) and pedestrians (adults, children, police) as well as static props (barrels, garbage cans, road barriers, *etc.*). The dynamic actors exhibited realistic movement patterns, governed by the underlying physics engine and adhered to real-world traffic rules, such as driving

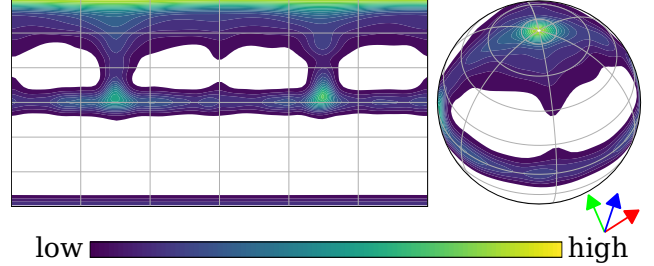low ─────────────────────── high

Figure 2. Spherical KDE plot using a von Mises-Fisher kernel to visualize surface normal distribution. Yellow regions indicate higher density of surface normals. White regions in the south correspond to physically impossible orientations, while those in the north represent extremely rare occurrences.

on designated roads and obeying traffic signals.

To capture realistic driving scenarios, we employ a virtual LiDAR sensor mounted atop a car operating in autopilot mode. The LiDAR sensor is configured to emit 64 laser beams, a $10°$ upper and a $-30°$ lower field of view. Such a common sensor configuration (*e.g.* [16, 48]) strikes a balance between sparsity and density, providing a challenging yet fair evaluation environment. To further mimic real-world conditions, we set the maximum range to 100 meters and introduce Gaussian noise with a standard deviation of 0.02 meters to the LiDAR point cloud. The sensor captures data at a rate of 10 Hz. We provide additional information about our simulation setup in the appendix.

CARLA's default LiDAR sensor implementation is limited to position and intensity channels. To enable surface normal collection, we extend CARLA's ray tracer to query surface normals at each intersection point between a ray and a mesh object. These surface normals are then transformed into the sensor's coordinate frame and appended to the Li-DAR data. This requires modifications to both CARLA's C++ backend and Python frontend, adding three extra channels to store the x, y, and z components of the normal vector for each LiDAR point.

For each map, we conduct eleven randomly initialized and independent simulation runs. A simulation is terminated early if prolonged traffic halts, such as red lights, occur. On average, each simulation lasts approximately 50 seconds, resulting in total of 50 045 labeled frames. To ensure rigorous evaluation, we partition our dataset into training, validation, and testing sets. We assign each map to exactly one split, preventing data leakage (*i.e.* using the same "city" in multiple splits). One map is designated for validation, while the remaining eight maps are divided equally between the training and testing sets. This results in 25 053 training, 22 167 testing, and 2825 validation frames. The full dataset will be released publicly via a Research Data Management platform upon publication.

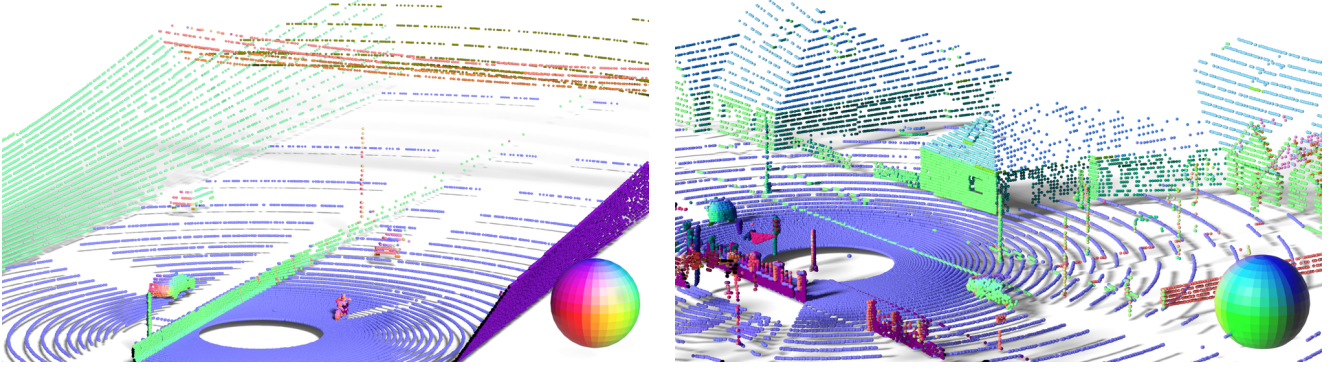Surface normals, being unit vectors, can be intuitively

Figure 3. Exemplary LiDAR frames from our LiSu dataset (left: tunnel portal, right: urban scene). Surface normals are linearly mapped to the RGB color space. The color legend spheres in the bottom right corners provide a visual reference to interpret the normal directions.

visualized as points on a unit sphere. Fig. 2 illustrates a spherical Kernel Density Estimate (KDE) plot using a von Mises-Fisher kernel to represent the distribution of surface normals. Our CARLA simulation, mirroring real-world environments, inherently produces imbalanced datasets. For instance, the southern hemisphere, excluding a minor region around the south pole corresponding to ceiling surfaces (*e.g.* in tunnels in Fig. 3), is largely unoccupied, since it is not physically possible for a surface to be oriented away from the LiDAR sensor. Additionally, real-world scenes are dominated by road surfaces (concentrated near the north pole) and wall surfaces (near the equator), leading to a highly skewed dataset. We address the challenges posed by this data imbalance in Sec. 4.1.

3.2. LiDAR Surface Normal Estimation

Given a LiDAR point cloud $\mathbf{P} \in \mathbb{R}^{N \times 3}$, we aim to learn a mapping $\Phi : \mathbb{R}^{N \times 3} \mapsto \mathbb{R}^{N \times 3}$ that, for each point $\mathbf{p}_i \in \mathbf{P}$, predicts a corresponding surface normal $\hat{\mathbf{n}}_i \in \mathbb{R}^3$. While our method is architecture agnostic, we employ state-of-the-art Point Transformer V3 (PTv3) [58] as Φ. Given the ground truth label $\mathbf{n}$, our loss function is defined as

$$\mathcal{L}_{L1} = \frac{1}{N} \sum_{i \in N} ||\mathbf{n}_i - \hat{\mathbf{n}}_i||_1. \quad (1)$$

While this type of supervision can guide the overall training procedure towards a local minima, it often fails to guarantee spatial and temporal coherence or accurate unit sphere estimates: As a result, the model may produce inconsistent predictions, such as assigning opposing normal directions to points on the same surface or generating significantly different surface normal estimates for the same surface across consecutive frames.

To overcome these limitations, we propose three additional data terms that exploit the following key observations: (1) normal vectors should be piecewise smooth, (2) normal vectors should exhibit temporal consistency, and (3) normal

vectors, being unit vectors, should be constrained to the unit sphere. To enforce both spatial and temporal coherence, we adopt Graph Total Variation (GTV) regularization. While our formulation shares similarities with prior works in point cloud super-resolution [11, 12], we deviate by incorporating GTV as a data term regularizer instead of the primary objective function. In the following, we delve deeper into our regularization terms and present the overall loss objective.

Spatial GTV (SGTV) Regularization. The spatial regularization term operates under the assumption that geometrically close LiDAR points belong to the same surface and thus share the same surface normal vector. We first convert our point cloud $\mathbf{P}$ into a k-neighborhood graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, with set of nodes $\mathcal{V}$ and edges $\mathcal{E}$. We associate each node $i \in \mathcal{V}$ with a location $\mathbf{p_i}$ and the respective surface normal prediction $\Phi(\mathbf{p}_i) = \hat{\mathbf{n}}_i$. The edge weights are defined with an adjacency matrix $\mathbf{W}_{ij} = [w_{ij}]$, where

$$w_{ij} = \exp\left(-\frac{||\mathbf{p}_i - \mathbf{p}_j||_2^2}{\sigma^2}\right) \quad (2)$$

considers the spatial proximity between points p_i and p_j. It assumes values close to 1 for nearby points and tends towards 0 as the distance between them grows. The parameter σ determines the scale at which proximity is measured, influencing the rate of decay of the edge weights with increasing distance. During optimization, we penalize spatially inconsistent surface normal predictions by

$$\mathcal{L}_{\text{SGTV}} = \frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} w_{ij} ||\hat{\mathbf{n}}_i - \hat{\mathbf{n}}_j||_1. \quad (3)$$

Temporal GTV (TGTV) Regularization. LiDAR point clouds are commonly captured sequentially at high frequency (*e.g.* 10 Hz as in ONCE [37], WOD [48], or Argoverse [57]), resulting in substantial overlap between consecutive LiDAR scans. Global registration modules, such as GPS coupled with IMU, allow for alignment of these frames to a common

4

key frame. This is particularly effective for static objects, while dynamic objects, such as moving vehicles, may exhibit slight misalignments. However, in typical autonomous driving settings, the number of LiDAR points corresponding to static scene contents vastly outnumbers the number of points corresponding to dynamic objects. We exploit the sequential nature of these datasets to enforce temporally coherent surface normal predictions.

During training, we employ mini-batches comprising pairs of consecutive point clouds, $(\mathbf{P}^t, \mathbf{P}^{t+1})$, along with their corresponding global poses (with respect to the key frame), $(\mathbf{T}^t, \mathbf{T}^{t+1}) \in \mathbb{R}^{4 \times 4}$. Both point clouds undergo random affine transformations, $(\mathbf{A}^t, \mathbf{A}^{t+1}) \in \mathbb{R}^{4 \times 4}$, encompassing rotations, translations, flips, and other standard augmentations. For improved readability, we adopt the convention $\mathbf{P} \triangleq \mathbf{ATP}$ throughout the paper.

After the forward pass $\Phi(\mathbf{P}^t)$ and $\Phi(\mathbf{P}^{t+1})$, we construct a bipartite graph $\mathcal{G} = (\{\mathcal{V}^t, \mathcal{V}^{t+1}\}, \mathcal{E}, \mathbf{W})$, where each node $i \in \mathcal{V}^t$ is associated with the transformed key frame point $(\mathbf{T}^t \mathbf{A}^t)^{-1} \mathbf{p}_i^t$ and key frame surface normal prediction $(\mathbf{T}^t \mathbf{A}^t)^{-1} \hat{\mathbf{n}}_i^t$. Similarly, each node $j \in \mathcal{V}^{t+1}$ is associated with the point $(\mathbf{T}^{t+1} \mathbf{A}^{t+1})^{-1} \mathbf{p}_j^{t+1}$ and surface normal prediction $(\mathbf{T}^{t+1} \mathbf{A}^{t+1})^{-1} \hat{\mathbf{n}}_j^{t+1}$. Analogous to Eq. (2), the weight adjacency matrix is defined as

$$w_{ij} = \exp\left(-\frac{\left\|(\mathbf{T}^t \mathbf{A}^t)^{-1} \mathbf{p}_i^t - (\mathbf{T}^{t+1} \mathbf{A}^{t+1})^{-1} \mathbf{p}_j^{t+1}\right\|_2^2}{\sigma^2}\right). \tag{4}$$

Finally, we penalize incoherence between two consecutive frames with

$$\mathcal{L}_{\text{TGTV}} = \frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} w_{ij} \|(\mathbf{T}^t \mathbf{A}^t)^{-1} \hat{\mathbf{n}}_j^t - (\mathbf{T}^{t+1} \mathbf{A}^{t+1})^{-1} \hat{\mathbf{n}}_j^{t+1}\|_1, \tag{5}$$

which enforces temporal smoothness.

Eikonal Regularization. Recent studies [18, 22, 53] have shown that enforcing unit-norm constraints on SDF gradients at object boundaries significantly improves model robustness. Following this principle, we constrain our predicted surface normals to lie on the unit sphere:

$$\mathcal{L}_{\text{Eikonal}} = \frac{1}{N} \sum_{i \in N} (\|\hat{\mathbf{n}}_i\|_2 - 1)^2. \tag{6}$$

Training Objective. Our training objective is a weighted combination of the loss and data regularization terms:

$$\mathcal{L} = \mathcal{L}_{L1} + \gamma(\mathcal{L}_{\text{SGTV}} + \mathcal{L}_{\text{TGTV}} + \mathcal{L}_{\text{Eikonal}}). \tag{7}$$

We empirically determined the optimal trade-off parameter γ to be 0.1.

4. Experiments

In the following, we present a comprehensive experimental evaluation demonstrating the superior performance of our method on LiSu for LiDAR surface normal estimation (Sec. 4.1). We further highlight the positive impact of our LiSu on direct transfer to real-world datasets (Sec. 4.2) and self-supervised domain adaptation. Finally, we show how our regularization terms mitigate the negative effects of noisy pseudo-labels (Sec. 4.2) in real-world neural surface reconstruction.

4.1. Surface Normal Evaluation

Dataset. Due to the lack of LiDAR datasets with surface normal annotations, in the following we conduct our experiments on our LiSu dataset. We train our models on the designated training split, comprising 25 053 labeled samples. Previous methods, including PCPNet [21] and SHS-Net [31], partition point clouds into N-point patches, necessitating multiple inference steps for each frame. These methods require on average over 10 seconds per frame, making comprehensive evaluation time-consuming. To enable a fair comparison within a reasonable timeframe, we downsampled our test set to every fifth frame, reducing the number of test frames to 4433. In contrast, our proposed method processes the entire point cloud in a single inference step. This allows us to process each frame in approximately 50 milliseconds.
Metric. Previous works on LiDAR surface normal estimation, such as [34], have adopted evaluation protocols inspired by PCPNet [21]. However, these protocols, which involve progressively adding artificial noise, are not ideal for inherently noisy LiDAR data. Other methods, like [45], impose overly strict error thresholds (*e.g.* $3°$), making them equally unsuitable. Instead, we propose a more realistic evaluation protocol that accounts for the natural noise and sparsity of LiDAR data. More specifically, we report the mean (**mean**), median (**median**) and root mean square (**RMSE**) angular error in degrees. Additionally, we provide accuracy metrics for various angular error thresholds: $\mathbf{5.0°}$, $\mathbf{7.5°}$, $\mathbf{11.25°}$, $\mathbf{22.5°}$, and $\mathbf{30.0°}$. These metrics indicate the percentage of predictions with angular error below the respective threshold.
Baseline. First, we benchmark our method against two unsupervised surface normal estimation methods: PCA [23] and Jet [7]. For both, we utilize a neighborhood size of $k = 32$ points and orient normals towards a common viewpoint. Subsequently, we train seven state-of-the-art supervised methods on the training split of our LiSu dataset: PCPNet [21], CMG-Net [59], GraphFit [28], Du *et al.* [14], NGL [30], NeuralGF [33], and SHS-Net [31]. To adapt to the characteristics of LiDAR data, we modify the original implementations by reducing the number of points per patch from 500 (within a 0.1 meter radius) to 32. To accommodate the two orders of magnitude increase in scale of the training data, we have opted to reduce the number of training epochs. This allows

Method	mean ↓	median ↓	RMSE ↓	5.0° ↑	7.5° ↑	11.25° ↑	22.5° ↑	30.0° ↑	Avg. Runtime
PCA [23]	76.18	29.10	110.95	44.95	46.86	48.45	51.02	52.09	**0.06**
Jet [7]	39.02	25.11	53.96	11.08	20.75	33.31	50.47	56.97	14.55
PCPNet [21]	9.88	1.42	20.56	67.42	72.41	76.87	84.75	88.10	14.31
CMG-Net [59]	10.05	2.44	20.24	65.15	71.09	75.94	78.40	80.03	40.01
GraphFit [28]	8.98	0.93	19.57	73.49	75.68	77.42	79.88	80.80	19.42
Du *et al.* [14]	8.81	0.93	19.54	73.53	75.92	77.75	80.05	80.88	16.26
NGL [30]	9.88	1.21	18.43	77.39	79.48	81.44	84.70	86.24	26.27
NeuralGF [33]	8.57	1.19	19.57	67.08	71.42	74.84	79.48	81.37	17.58
SHS-Net [31]	<u>6.46</u>	<u>0.67</u>	<u>18.40</u>	<u>79.66</u>	<u>83.57</u>	<u>86.77</u>	<u>90.79</u>	<u>92.06</u>	24.27
Ours	**6.30**	**0.43**	**17.58**	**81.10**	**84.44**	**87.32**	**91.66**	**93.09**	<u>0.09</u>

Table 2. Evaluation of surface normal estimation methods on the LiSu test split. Mean, median, and root mean square (RMSE) angular error in degrees, as well as accuracy for various angular error thresholds, and average runtime per LiDAR point cloud (in seconds) are reported. **Best** and <u>second-best</u> values are highlighted in bold and underlined, respectively.

us to maintain the same total number of iterations, ensuring that the model is exposed to an equivalent amount of training data. We provide more detailed information in the appendix.

Implementation Details. We adopt the default PTv3 [58] configuration, which consists of a four-stage encoder-decoder architecture. We optimize our model using AdamW [36] with a maximum learning rate of 0.003 and a weight decay of 0.005. The learning rate schedule comprises a short (1 epoch) warm-up phase followed by cosine annealing [35]. We train our model on 8 A100 GPUs using mixed precision with a batch size of 32. Standard point cloud augmentation techniques, such as rotation, flipping, and scaling, are utilized. Training is terminated after 50 epochs.

To address the class imbalance in our dataset (illustrated in Fig. 2), we employ a weighted loss function: Each data sample is weighted inversely proportional to the frequency of its ground truth surface normal occurrence. This effectively upweights underrepresented classes, such as less common surface orientations, while downweighting overrepresented classes. Consequently, the model is encouraged to focus more on the challenging, less frequent samples.

Results. Table 2 presents the comparison of the surface normal estimation methods. Traditional methods, such as PCA [23] and Jet [7], are known to be sensitive to noise and prone to ambiguities in the orientation of their surface normal estimates. When applied to sparse and non-uniform LiDAR data, as demonstrated in Tab. 2, these methods exhibit significant limitations, often producing unreliable results.

Our proposed training method allows us to build a PTv3-based surface normal estimator that comfortably surpasses state-of-the-art supervised methods in both accuracy and speed. Compared to PCPNet [21], GraphFit [28], and Du *et al.* [14], we achieve a significant reduction of over 2.5° in mean average angular error and an 8% increase in accuracy at the 5.0° threshold. Additionally, we outperform SHS-Net [31] by reducing the median angular error by 0.24° and

$\mathcal{L}_{L1}$	$\mathcal{L}_{\text{SGTV}}$	$\mathcal{L}_{\text{TGTV}}$	$\mathcal{L}_{\text{Eikonal}}$	RMSE ↓	5.0° ↑
✓				19.82	78.70
✓	✓			17.78	81.06
✓		✓		<u>17.72</u>	<u>81.08</u>
✓			✓	17.74	80.65
✓	✓	✓	✓	**17.58**	**81.10**

Table 3. Influence of proposed components on the in-domain model performance. Models are trained on the full LiSu training split and evaluated on a 20% test split.

improve the 5.0° accuracy by 1.44°. Furthermore, our model processes a single LiSu frame in just 90 milliseconds on average, significantly outperforming state-of-the-art methods like SHS-Net, PCPNet, and GraphFit (which require over 14 seconds). This substantial speedup is attributed to our approach of processing the entire LiDAR point cloud at once, eliminating the need for partitioning and multiple inference steps, as required by state-of-the-art methods.

Regularization Ablation Study. To assess the individual contribution of each component proposed in Sec. 3.2, we conduct an ablation study shown in Tab. 3. Each component, when trained from scratch on our full training split, outperforms our baseline model (trained with $\mathcal{L}_{L1}$ loss only). The model incorporating all proposed losses demonstrates the best overall performance.

4.2. Synthetic-to-Real Domain Adaptation

In the following, we qualitatively assess the state-of-the-art in surface normal estimation on real-world LiDAR point clouds from the Waymo Open Dataset [48]. We highlight the limitations of traditional methods and show that direct transfer from our LiSu dataset significantly reduces the synthetic-to-real domain gap compared to the existing PCPNet [21] dataset. Furthermore, we demonstrate how our method can effectively bridge this gap.

Classical approaches based on PCA [23] (see Figure 4, first row) do not require labeled data but suffer from surface normal orientation ambiguities, particularly with noisy LiDAR point clouds. For instance, points on a road with upward (blue) or downward (brown) normals can be interpreted as correct, depending on the perspective. Yet, physical constraints dictate a single correct orientation, which is upward in this case. More recent approaches, such as SHS-Net [31] (second row in Figure 4), trained on the PCPNet dataset, exhibit similar limitations while providing smoother predictions. Our direct domain transfer experiments (third and forth row in Fig. 4), *i.e.* models trained on our LiSu dataset and employed without any adaptation to Waymo, reveal already tremendous improvement. This demonstrates that for the downstream task of LiDAR surface normal estimation, our LiSu dataset, as anticipated, is much better suited than existing CAD-based datasets like PCPNet.

To further enhance model robustness and generalization, we leverage our surface estimation method (Sec. 3.2) within a self-training paradigm, successfully reducing the domain gap. We first pretrain our model on the LiSu training split, followed by a second phase on the Waymo train split, leveraging pseudo-labels and our proposed data term regularization to mitigate noise. In the second (self-training) phase, we optimize our model using AdamW [36] with a maximum learning rate of 0.001 and weight decay of 0.005. To preserve the learned feature representation from the clean data, we employ differential learning rates [62], assigning lower learning rates to shallow layers and exponentially decaying them with network depth by a factor of 0.85. We employ standard point cloud augmentation techniques, such as rotation, flipping, and scaling. The learning rate schedule employs a short warm-up phase followed by linear annealing [35]. Our model is trained on 4 A100 GPUs using mixed precision with a batch size of 32 for 10 epochs. In contrast to other approaches, our method consistently produces accurate, smooth, and correctly oriented surface normal predictions.

4.3. Neural Surface Reconstruction

Our work demonstrates that the combination of our method with LiSu effectively reduces the domain gap between datasets. However, the specific influence of our regularization technique on the overall self-training process remains unclear. Due to the scarcity of real-world LiDAR datasets with surface normal annotations, we opted to evaluate its effectiveness indirectly through a downstream task: neural surface reconstruction.

Previous methods in this domain often leverage geometric priors, such as surface normals derived from monocular images [19, 22, 52], to improve reconstruction quality. These methods typically penalize discrepancies between the SDF boundary gradients and the predicted surface normals [63]. However, LiDAR-only neural surface reconstruc-
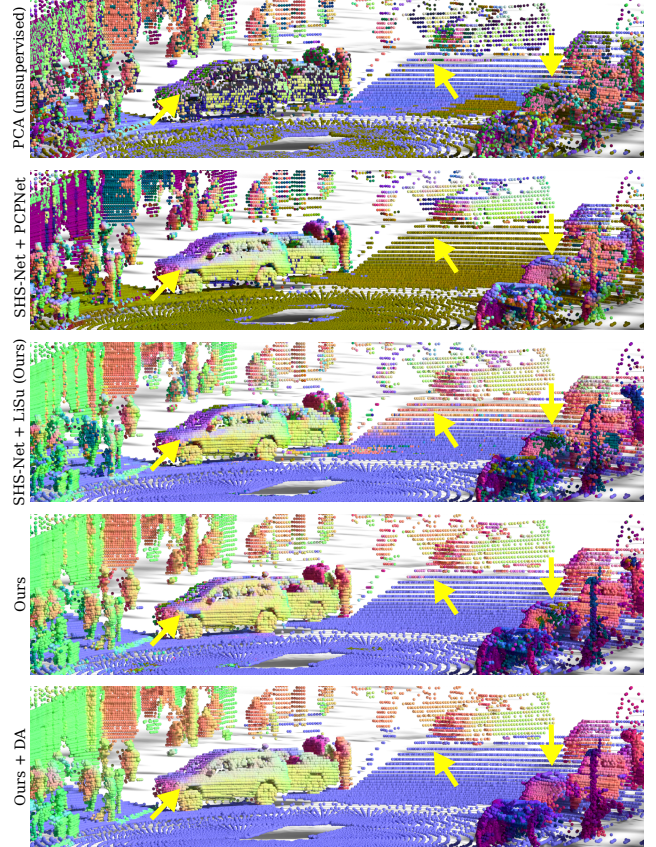

Figure 4. Qualitative evaluation of traditional and learning-based methods on a challenging Waymo frame. Notably, the current state-of-the-art (SHS-Net [31] trained on PCPNet [21]) struggles to generalize to noisy and sparse LiDAR data. When trained on our proposed dataset (LiSu), both SHS-Net and our method yield reasonable results on the Waymo frame. In the last row, we demonstrate how leveraging our method within a self-supervised learning paradigm significantly enhances overall estimation, resulting in accurate, smooth, and consistently oriented predictions.

tion approaches, such as ReSimAD [66], are limited by the absence of reliable surface normal estimation methods for real-world LiDAR data. Therefore, we leverage our unsupervised training approach from Sec. 4.2, to learn a robust LiDAR surface normal estimator from the real-world Waymo dataset. This prior is then integrated into ReSimAD to enhance its surface reconstruction capabilities.

We inherit the evaluation protocol established in [22, 66]: For each sequence, we train a SDF model using point cloud data from all five Waymo LiDAR sensors, as described in [66]. A single mesh is then reconstructed for the entire sequence using this trained model. To simulate a real-world top-mounted LiDAR, virtual sensors are positioned at the locations of the real physical sensors and synthetic point clouds are generated by ray-casting the reconstructed mesh. We evaluate the accuracy of these synthetic point clouds

Seq.	ReSimAD [66]	DT	$\mathcal{L}_{\mathbf{L1}}$	$\mathcal{L}$
1006130	2.53 / 0.25	**2.51** / 0.26	**2.51** / 0.24	**2.51** / **0.23**
1172406	1.95 / 0.52	1.95 / 0.50	1.96 / 0.51	**1.90** / **0.47**
1323841	2.16 / 0.70	2.15 / 0.75	2.16 / 0.74	**2.12** / **0.68**
1347637	2.07 / 0.33	2.05 / 0.25	2.05 / 0.29	**2.02** / 0.28
1486973	1.80 / **0.19**	1.83 / 0.21	**1.79** / 0.20	**1.79** / 0.20
1506235	1.54 / 0.38	1.50 / 0.38	1.52 / 0.37	**1.46** / **0.34**
1664636	2.15 / 0.53	2.19 / 0.51	2.10 / 0.48	**2.08** / **0.47**
4058410	1.91 / 0.36	1.91 / 0.34	1.91 / 0.40	**1.90** / **0.31**
average	2.01 / 0.41	2.01 / 0.40	2.00 / 0.40	**1.97** / **0.37**

Table 4. We evaluate neural surface reconstruction on diverse Waymo sequences using Root Mean Square Error (RMSE) / Chamfer Distance (CD). Lower values indicate better performance. ReSimAD [66] omits surface normal loss during reconstruction. Direct Transfer (DT) applies a model trained on LiSu without adaptation. $\mathcal{L}$ (Eq. (7)) and $\mathcal{L}_{\mathbf{L1}}$ (Eq. (1)) denote Waymo self-trained models with and without regularization, respectively.

by comparing them to the corresponding real-world point clouds using Root Mean Square Error (RMSE) and Chamfer Distance (CD) metrics in Table 4.

Surface reconstruction in ReSimAD is solely guided by the LiDAR point cloud information. While this already produces reasonably good meshes, the SDF of regions that are not hit by a LiDAR beam remain undefined. This leads to coarse meshes and jittering artifacts clearly visible in the exemplary ReSimAD results in Fig. 5. Incorporating estimated surface normals into ReSimAD improves the mesh reconstruction notably, as illustrated in the three bottom rows of Fig. 5: Directly transfering (DT) a surface normal estimator trained (solely) on LiSu to the Waymo data results in better surface normal estimates and, consequently, a smoother surface reconstruction near the ego vehicle. Subsequently applying our proposed self-supervised domain adaption further improves the results: the bottom row of Fig. 5 illustrates the benefits of our proposed regularization, which leads to smooth surface meshes, even at far distances from the ego vehicle. Naïvely performing self-supervised adaptation without our regularization (*i.e.* solely using the $\mathcal{L}_{L1}$ loss from Eq. (1)), on the other hand, does not improve the results.

5. Conclusion

We introduced LiSu, a synthetic dataset designed to advance research on LiDAR-based surface normal estimation. With more than 50 000 LiDAR frames, obtained from a state-of-the-art traffic simulation engine, LiSu presents an ideal testbed to develop and benchmark LiDAR surface normal estimation approaches. To properly leverage this large-scale synthetic data we propose a method that exploits the spatiotemporal nature of LiDAR scans in autonomous driving settings via meticulously designed regularization constraints. Moreover, we show that our method can be seamlessly extended to domain adaptation scenarios, where it helps to
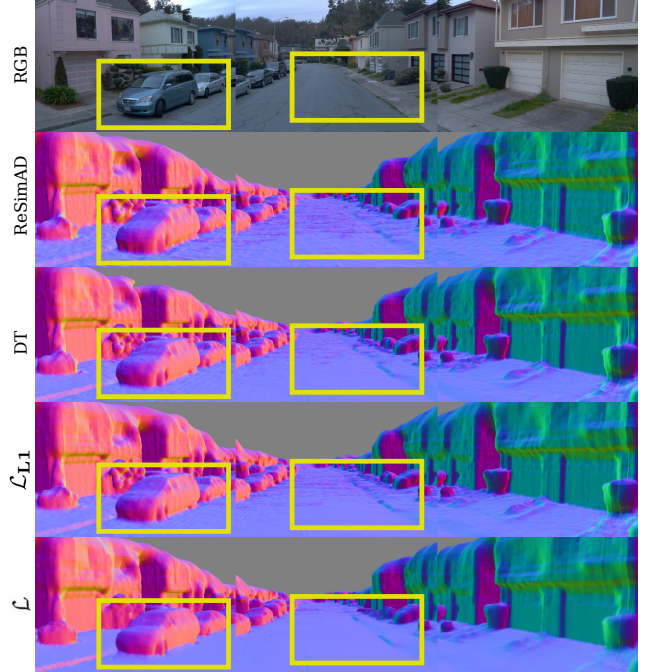

Figure 5. Reconstructed meshes from Waymo sequence 1172406, color-coded with surface normals. ReSimAD [66] omits surface normal loss during reconstruction. Direct Transfer (DT) applies a model trained on LiSu without adaptation. $\mathcal{L}$ (Eq. (7)) and $\mathcal{L}_{\mathbf{L1}}$ (Eq. (1)) denote Waymo self-trained models with and without regularization, respectively.

alleviate the negative effects of noisy pseudo-labels. Across several experiments, we demonstrate the utility of LiSu and our method to obtain outstanding results in surface normal estimation and their positive impact on the downstream task of surface reconstruction from real-world LiDAR point clouds. We believe the LiSu dataset will help our research community to improve LiDAR-based surface normal estimation approaches and, consequently, leverage these to advance autonomous driving: For example, scenario generation for domain adaptation research (*i.e.*, re-rendering scenes with different simulated LiDAR configurations) will benefit from improved surface reconstructions.

References

[1] H. Badino, D. Huber, Y. Park, and T. Kanade. Fast and Accurate Computation of Surface Normals from Range Images. In *Proc. ICRA*, 2011. 2

[2] Gwangbin Bae and Andrew J. Davison. Rethinking Inductive

Biases for Surface Normal Estimation. In *Proc. CVPR*, 2024. 1

[3] Jens Behley and Cyrill Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. RSS*, 2018. 1

[4] Yizhak Ben-Shabat and Stephen Gould. DeepFit: 3D Surface Fitting via Neural Network Weighted Least Squares. In *Proc. ECCV*, 2020. 2

[5] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. Nesti-Net: Normal Estimation for Unstructured 3D Point Clouds Using Convolutional Neural Networks. In *Proc. CVPR*, 2019. 2

[6] Igor Bogoslavskyi, Konstantinos Zampogiannis, and Raymond Phan. Fast and Robust Normal Estimation for Sparse LiDAR Scans. In *Proc. ICRA*, 2024. 2

[7] F. Cazals and M. Pouget. Estimating Differential Quantities Using Polynomial Fitting of Osculating Jets. *CAGD*, 22(2): 121–146, 2005. 2, 5, 6

[8] Zhuoxiao Chen, Yadan Luo, Zheng Wang, Mahsa Baktashmotlagh, and Zi Huang. Revisiting Domain-Adaptive 3D Object Detection by Reliable, Diverse and Class-balanced Pseudo-Labeling. In *Proc. ICCV*, 2023. 2

[9] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *Proc. ICLR*, 2024. 12

[10] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Proc. NeurIPS*, 2022. 12

[11] Chinthaka Dinesh, Gene Cheung, and Ivan V. Bajić. 3D Point Cloud Super-Resolution via Graph Total Variation on Surface Normals. In *Proc. ICIP*, 2019. 4

[12] Chinthaka Dinesh, Gene Cheung, and Ivan V Bajić. Super-Resolution of 3D Color Point Clouds Via Fast Graph Total Variation. In *Proc. ICASSP*, 2020. 4

[13] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In *Proc. CoRL*, 2017. 1, 3, 16

[14] Hang Du, Xuejun Yan, Jingjing Wang, Di Xie, and Shiliang Pu. Rethinking the Approximation Error in 3D Surface Fitting for Point Cloud Normal Estimation. In *Proc. CVPR*, 2023. 1, 2, 5, 6, 12, 13

[15] Rui Fan, Hengli Wang, Peide Cai, and Ming Liu. SNE-RoadSeg: Incorporating Surface Normal Information into Semantic Segmentation for Accurate Freespace Detection. In *Proc. ECCV*, 2020. 1

[16] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. CVPR*, 2012. 2, 3

[17] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets Robotics: The KITTI Dataset. *IJRR*, 32(11):1231–1237, 2013. 12

[18] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit Geometric Regularization for Learning Shapes. In *Proc. ICML*, 2020. 5

[19] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit Geometric Regularization for Learning Shapes. In *Proc. PMLR*, 2020. 3, 7

[20] Gael Guennebaud and Markus Gross. Algebraic Point Set Surfaces. In *Proc. SIGGRAPH*, 2007. 2

[21] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. PCPNet Learning Local Shape Properties from Raw Point Clouds. *CGF*, 37(2):75–85, 2018. 1, 2, 5, 6, 7, 12, 13, 16, 17

[22] Jianfei Guo, Nianchen Deng, Xinyang Li, Yeqi Bai, Botian Shi, Chiyu Wang, Chenjing Ding, Dongliang Wang, and Yikang Li. StreetSurf: Extending Multi-view Implicit Surface Reconstruction to Street Views. *arXiv CoRR*, abs/2306.04988, 2023. 2, 3, 5, 7, 13

[23] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proc. SIGGRAPH*, 1992. 2, 5, 6, 7, 12

[24] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3D v2: A Versatile Monocular Geometric Foundation Model for Zero-shot Metric Depth and Surface Normal Estimation. *arXiv CoRR*, abs/2404.15506, 2024. 1

[25] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In *Proc. SGP*, 2006. 3

[26] Rawal Khirodkar, Timur Bagautdinov, Julieta Martinez, Su Zhaoen, Austin James, Peter Selednik, Stuart Anderson, and Shunsuke Saito. Sapiens: Foundation for Human Vision Models. In *Proc. ECCV*, 2024. 1

[27] Jan Eric Lenssen, Christian Osendorfer, and Jonathan Masci. Deep iterative surface normal estimation. In *Proc. CVPR*, 2020. 2

[28] Keqiang Li, Mingyang Zhao, Huaiyu Wu, Dong-Ming Yan, Zhen Shen, Fei-Yue Wang, and Gang Xiong. GraphFit: Learning Multi-scale Graph-Convolutional Representation for Point Cloud Normal Estimation. In *Proc. ECCV*, 2022. 2, 5, 6, 12, 13

[29] Qing Li, Yu-Shen Liu, Jin-San Cheng, Cheng Wang, Yi Fang, and Zhizhong Han. HSurf-Net: Normal Estimation for 3D Point Clouds by Learning Hyper Surfaces. In *Proc. NeurIPS*, 2022. 2

[30] Qing Li, Huifang Feng, Kanle Shi, Yi Fang, Yu-Shen Liu, and Zhizhong Han. Neural Gradient Learning and Optimization for Oriented Point Normal Estimation. In *Proc. SIGGRAPH Asia*, 2023. 5, 6

[31] Qing Li, Huifang Feng, Kanle Shi, Yue Gao, Yi Fang, Yu-Shen Liu, and Zhizhong Han. SHS-Net: Learning Signed Hyper Surfaces for Oriented Normal Estimation of Point Clouds. In *Proc. CVPR*, 2023. 1, 5, 6, 7, 12, 13, 16, 17

[32] Qing Li, Huifang Feng, Kanle Shi, Yue Gao, Yi Fang, Yu-Shen Liu, and Zhizhong Han. Learning signed hyper surfaces for oriented point cloud normal estimation. *TPAMI*, 2024. 1

[33] Qing Li, Huifang Feng, Kanle Shi, Yue Gao, Yi Fang, Yu-Shen Liu, and Zhizhong Han. NeuralGF: Unsupervised Point Normal Estimation by Learning Neural Gradient Function. In *Proc. NeurIPS*, 2024. 1, 5, 6

[34] Ancheng Lin, Jun Li, Yusheng Xiang, Wei Bian, and Mukesh Prasad. Normal Transformer: Extracting Surface Geometry from LiDAR Points Enhanced by Visual Semantics. In *Proc. IEEE T-IV*, 2024. 2, 3, 5

[35] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *Proc. ICLR*, 2017. 6, 7

[36] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *Proc. ICLR*, 2019. 6, 7

[37] Jiageng Mao, Minzhe Niu, Chenhan Jiang, Hanxue Liang, Jingheng Chen, Xiaodan Liang, Yamin Li, Chaoqiang Ye, Wei Zhang, Zhenguo Li, Jie Yu, Hang Xu, and Chunjing Xu. One Million Scenes for Autonomous Driving: ONCE Dataset. In *Proc. NeurIPS*, 2021. 4

[38] Jishu Miao, Tsubasa Hirakawa, Takayoshi Yamashita, and Hironobu Fujiyoshi. 3D Object Detection with Normal-map on Point Clouds. In *Proc. VISIGRAPP*, 2021. 1

[39] Björn Michele, Alexandre Boulch, Gilles Puy, Tuan-Hung Vu, Renaud Marlet, and Nicolas Courty. SALUDA: Surface-based Automotive Lidar Unsupervised Domain Adaptation. In *Proc. 3DV*, 2024. 2

[40] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Trans. Robot.*, 33(5):1255–1262, 2017. 1

[41] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape As Points: A Differentiable Poisson Solver. In *Proc. NeurIPS*, 2021. 3

[42] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. CVPR*, 2017. 1, 2, 12

[43] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys. DeepLiDAR: Deep Surface Normal Guided Depth Prediction for Outdoor Scene From Sparse LiDAR Data and Single Color Image. In *Proc. CVPR*, 2019. 2

[44] Cristiano Saltori, Evgeny Krivosheev, Stéphane Lathuilière, Nicu Sebe, Fabio Galasso, Giuseppe Fiameni, Elisa Ricci, and Fabio Poiesi. GIPSO: Geometrically Informed Propagation for Online Adaptation in 3D LiDAR Segmentation. In *Proc. ECCV*. arXiv, 2022. 2

[45] Dominik Scheuble, Chenyang Lei, Seung-Hwan Baek, Mario Bijelic, and Felix Heide. Polarization Wavefront Lidar: Learning Large Scene Reconstruction from Polarized Wavefronts. In *Proc. CVPR*, 2024. 2, 3, 5

[46] Thomas Schops, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle Adjusted Direct RGB-D SLAM. In *Proc. CVPR*, 2019. 1

[47] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *Proc. NeurIPS*, 2020. 3

[48] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proc. CVPR*, 2020. 1, 2, 3, 4, 6, 12, 14, 15, 16

[49] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 3

[50] Xiaoyu Tian, Haoxi Ran, Yue Wang, and Hang Zhao. GeoMAE: Masked Geometric Target Prediction for Self-supervised Point Cloud Pre-Training. In *Proc. CVPR*, 2023. 1

[51] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields. In *Proc. CVPR*, 2022. 1

[52] Jiepeng Wang, Peng Wang, Xiaoxiao Long, Christian Theobalt, Taku Komura, Lingjie Liu, and Wenping Wang. NeuRIS: Neural Reconstruction of Indoor Scenes Using Normal Priors. In *Proc. ECCV*, 2022. 2, 3, 7

[53] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In *Proc. NeurIPS*, 2021. 5

[54] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In *Proc. NeurIPS*, 2021. 2, 3

[55] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic Graph CNN for Learning on Point Clouds. *TOG*, 38(5):1–12, 2019. 12

[56] Zian Wang, Tianchang Shen, Jun Gao, Shengyu Huang, Jacob Munkberg, Jon Hasselgren, Zan Gojcic, Wenzheng Chen, and Sanja Fidler. Neural Fields meet Explicit Geometric Representations for Inverse Rendering of Urban Scenes. In *Proc. CVPR*, 2023. 1

[57] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting. In *Proc. NeurIPS*, 2021. 4

[58] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point Transformer V3: Simpler, Faster, Stronger. In *Proc. CVPR*, 2024. 2, 4, 6, 12

[59] Yingrui Wu, Mingyang Zhao, Keqiang Li, Weize Quan, Tianqi Yu, Jianfeng Yang, Xiaohong Jia, and Dong-Ming Yan. CMG-Net: Robust Normal Estimation for Point Clouds via Chamfer Normal Distance and Multi-Scale Geometry. In *Proc. AAAI*, 2024. 5, 6

[60] Aoran Xiao, Jiaxing Huang, Dayan Guan, Fangneng Zhan, and Shijian Lu. Transfer Learning from Synthetic to Real LiDAR Point Cloud for Semantic Segmentation. In *Proc. AAAI*, 2022. 2

[61] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. ST3D++: Denoised Self-Training for Unsupervised Domain Adaptation on 3D Object Detection. *TPAMI*, 45(5):6354–6371, 2023. 2

[62] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized

Autoregressive Pretraining for Language Understanding. In *Proc. NeurIPS*, 2019. 7

[63] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction. In *Proc. NeurIPS*, 2022. 1, 2, 3, 7

[64] Zhimin Yuan, Wankang Zeng, Yanfei Su, Weiquan Liu, Ming Cheng, Yulan Guo, and Cheng Wang. Density-guided Translator Boosts Synthetic-to-Real Unsupervised Domain Adaptive Segmentation of 3D Point Clouds. In *Proc. CVPR*, 2024. 2

[65] Guangyao Zhai, Dianye Huang, Shun-Cheng Wu, HyunJun Jung, Yan Di, Fabian Manhardt, Federico Tombari, Nassir Navab, and Benjamin Busam. MonoGraspNet: 6-DoF Grasping with a Single RGB Image. In *Proc. ICRA*, 2023. 1

[66] Bo Zhang, Xinyu Cai, Jiakang Yuan, Donglin Yang, Jianfei Guo, Xiangchao Yan, Renqiu Xia, Botian Shi, Min Dou, Tao Chen, Si Liu, Junchi Yan, and Yu Qiao. ReSimAD: Zero-Shot 3D Domain Transfer for Autonomous Driving with Source Reconstruction and Target Simulation. In *Proc. ICLR*, 2024. 2, 3, 7, 8, 13, 14, 15, 16

[67] Weichen Zhang, Wen Li, and Dong Xu. SRDAN: Scale-aware and Range-aware Domain Adaptation Network for Cross-dataset 3D Object Detection. In *Proc. CVPR*, 2021. 2

[68] Zhanwei Zhang, Minghao Chen, Shuai Xiao, Liang Peng, Hengjia Li, Binbin Lin, Ping Li, Wenxiao Wang, Boxi Wu, and Deng Cai. Pseudo Label Refinery for Unsupervised Domain Adaptation on Cross-Dataset 3D Object Detection. In *Proc. CVPR*, 2024. 2

[69] Haimei Zhao, Jing Zhang, Zhuo Chen, Shanshan Zhao, and Dacheng Tao. UniMix: Towards Domain Adaptive and Generalizable LiDAR Semantic Segmentation in Adverse Weather. In *Proc. CVPR*, 2024. 2

[70] Sicheng Zhao, Yezhen Wang, Bo Li, Bichen Wu, Yang Gao, Pengfei Xu, Trevor Darrell, and Kurt Keutzer. ePointDA: An End-to-End Simulation-to-Real Domain Adaptation Framework for LiDAR Point Cloud Segmentation. In *Proc. AAAI*, 2021. 2

[71] Haoran Zhou, Honghua Chen, Yingkui Zhang, Mingqiang Wei, Haoran Xie, Jun Wang, Tong Lu, Jing Qin, and Xiao-Ping Zhang. Refine-net: Normal refinement neural network for noisy point clouds. *TPAMI*, 45(1):946–963, 2022. 1

[72] Runsong Zhu, Yuan Liu, Zhen Dong, Yuan Wang, Tengping Jiang, Wenping Wang, and Bisheng Yang. AdaFit: Rethinking learning-based normal estimation on point clouds. In *Proc. ICCV*, pages 6118–6127, 2021. 2

LiSu: A Dataset and Method for LiDAR Surface Normal Estimation

Supplementary Material

This supplementary material provides an in-depth analysis of inference speed for the models used in our benchmarks (App. A). Additionally, it offers comprehensive details on the acquisition of our LiSu dataset (App. B) and the implementation specifics of baseline methods (App. C). Furthermore, we present additional experiments for the neural surface reconstruction downstream task (App. D), qualitative evaluations (App. E), and a rigorous ablation study exploring the impact of various design choices (App. F, App. G, App. H).

A. Inference Speed *vs*. Accuracy

Traditional methods like PCA [23] are renowned for their efficient runtime. However, their accuracy is degraded by inherent rotation ambiguities. This often manifests as points on the same plane exhibiting opposite surface normal directions (recall Fig. 4 of the main manuscript). Common heuristics, such as orienting all normals towards a fixed viewpoint or propagating orientation information via Minimum Spanning Tree (MST) [23], alleviate this problem but, due to the noisy nature of LiDAR point clouds, are not particularly effective.

Supervised methods like SHS-Net [31], Du *et al*. [14], GraphFit [28], and PCPNet [21] offer substantial performance gains, but at the cost of significant computational overhead. A key limitation of these methods is their reliance on point cloud partitioning, required during both training and inference. Furthermore, they often employ point-based backbone architectures like PointNet [42] or architectures which require special operations such as DGCNN [55], which hinder efficient processing. Originally introduced to address the limited dataset size of PCPNet [21] (30 samples), point cloud partitioning remains necessary during inference, significantly increasing processing time, especially for large-scale datasets like ours, LiSu (approximately 100k points per frame). Batching partitions is a potential strategy for accelerating inference. However, GPU VRAM limits batch sizes, preventing single-frame inference and necessitating multiple inference passes for batched partitions. Moreover, PointNet and DGCNN are not optimized for large-scale point clouds, making it challenging to adapt them to single-frame training/inference.

Conversely, we leverage the Point Transformer V3 (PTv3) [58], a state-of-the-art transformer architecture build for large-scale LiDAR point clouds. Their employment of space-filling curves (*e.g*. z-order or Hilbert curve) for point cloud serialization and hardware-optimized operations (*e.g*. FlashAttention [9, 10]) enable significant speedups. A single inference step on a large-scale LiDAR point cloud can be executed in orders of magnitude less time (*e.g*. 50 milliseconds
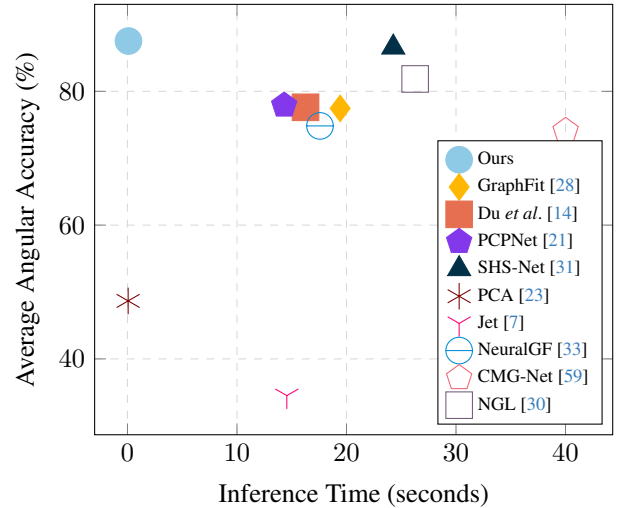

Figure 6. Inference speed *vs*. accuracy plot for various neural network-based and traditional methods. Accuracy is calculated as the average angular accuracy across all thresholds listed in Tab. 2 of the main manuscript: $\{5.0°, 7.5°, 11.25°, 22.5°, 30.0°\}$.

vs. 20 seconds for DGCNN). PTv3 coupled with our novel LiSu and training method exhibits exceptional performance in LiDAR surface normal estimation, requiring significantly less computational time compared to existing methods, as visualized in Fig. 6.

B. LiSu Acquisition

To obtain our custom LiSu dataset, we extended the CARLA simulator (version 0.9.15[1]) built with Unreal Engine 4.26[2] with an additional LiDAR sensor. This sensor captures not only standard position and intensity data but also the surface normal vector of each hit point. We extended CARLA's ray caster to return surface normals in the sensor's frame of reference and stream this data from the C++ core. This data stream was collect by the Python front-end and saved to a file together with the global sensor position and orientation (required for keyframe transformation in TGTV in Sec. 3.2 of the main manuscript)

We initialize a virtual LiDAR sensor to reflect commonly employed real-world LiDARs [17, 48]. A detailed configuration of this sensor is presented in Table 5. The virtual LiDAR sensor was mounted on a self-driving car, randomly placed within the simulation environment. We populated the simula-
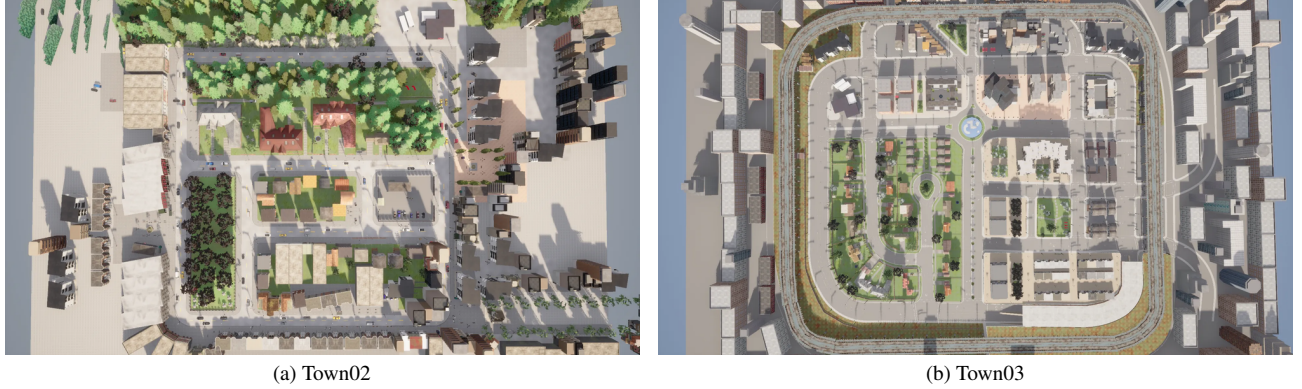
---

[1] https://github.com/carla-simulator/carla/releases/tag/0.9.15
[2] https://github.com/CarlaUnreal/UnrealEngine

(a) Town02	(b) Town03

Figure 7. Bird's eye view images of CARLA towns. Images taken from https://carla.readthedocs.io/en/latest/core_map/#carla-maps

Description	Value
Number of lasers	64
Maximum distance to raycast in meters	100
Points generated by all lasers per second	2M
LIDAR rotation frequency	10 Hz
Angle of the highest beam	$10°$
Angle of the lowest beam	$-30°$
Horizontal field of view	$360°$
Proportion of randomly dropped points	0.45
Std Dev point noise along the raycast vector	0.02

Table 5. Virtual LiDAR attributes used for the data acquisition.

tion environment with approximately 6000 dynamic actors, such as vehicles (cars, trucks, buses, vans, motorcycles, bicycles) and pedestrians (adults, children, police) as well as 2000 static props (barrels, garbage cans, road barriers, *etc*.). Due to the different map sizes (*e.g.* Fig. 7a *vs*. Fig. 7b), not all objects were guaranteed to appear in every simulation. For each map, we conducted N independent runs, each initialized with a different random seed. To avoid redundant frames, we terminated simulations prematurely if prolonged traffic halts, such as those caused by red lights, occurred. A detailed breakdown of simulation runs is provided in Tab. 6.

C. Implementation Details of Other Methods

We benchmarked our method against several state-of-the-art point cloud surface normal estimation methods: PCP-Net [21], GraphFit [28], Du *et al*. [14], and SHS-Net [31]. These methods share a common training strategy, originally proposed in PCPNet, which involves randomly sampling query points and their k-nearest neighbors from each training sample. While PCPNet's dataset consists of dense point clouds, our LiDAR data is significantly sparser. Consequently, we reduced the neighborhood size k to 32 to better

adapt to the sparse nature of our data. Furthermore, given our larger dataset, we decreased the number of training epochs to 10 for all methods. All other hyperparameters were kept consistent with their original settings.

D. Neural Surface Reconstruction

Our experiments in Sec. 4.3 closely replicate existing benchmarks in neural surface reconstruction from LiDAR data [22, 66]. In these benchmarks, a mesh reconstructed from all available LiDAR data is queried with rays generated from the same data, and the resulting distances are compared to actual LiDAR measurements. While this approach offers a convenient evaluation framework, it may not accurately reflect the method's true performance, as the same data is used both in training and evaluation.

Therefore, we propose a more rigorous evaluation protocol. We randomly split LiDAR points from an entire sequence into two disjoint sets: a training and a testing set. The training split is used exclusively in the training phase. Subsequently, the reconstructed mesh is evaluated using the unseen testing set. By ensuring that the training and testing sets are mutually exclusive, we can better identify model's potential limitations.

The proposed evaluation protocol proves particularly challenging for plain ReSimAD [66], as evident from the mesh reconstruction in Fig. 8 and LiDAR simulation in Fig. 9. Limited training data hinders SDF generalization, leading to poor extrapolation in areas lacking ground truth signals, such as ridges in the mesh (Fig. 8). This noise propagates to the LiDAR simulation (Fig. 9), resulting in highly noisy point clouds. Incorporating surface normals as an additional training signal mitigates these issues, leading to smoother meshes and consequently cleaner point clouds. This improvement is also quantified in Tab. 7
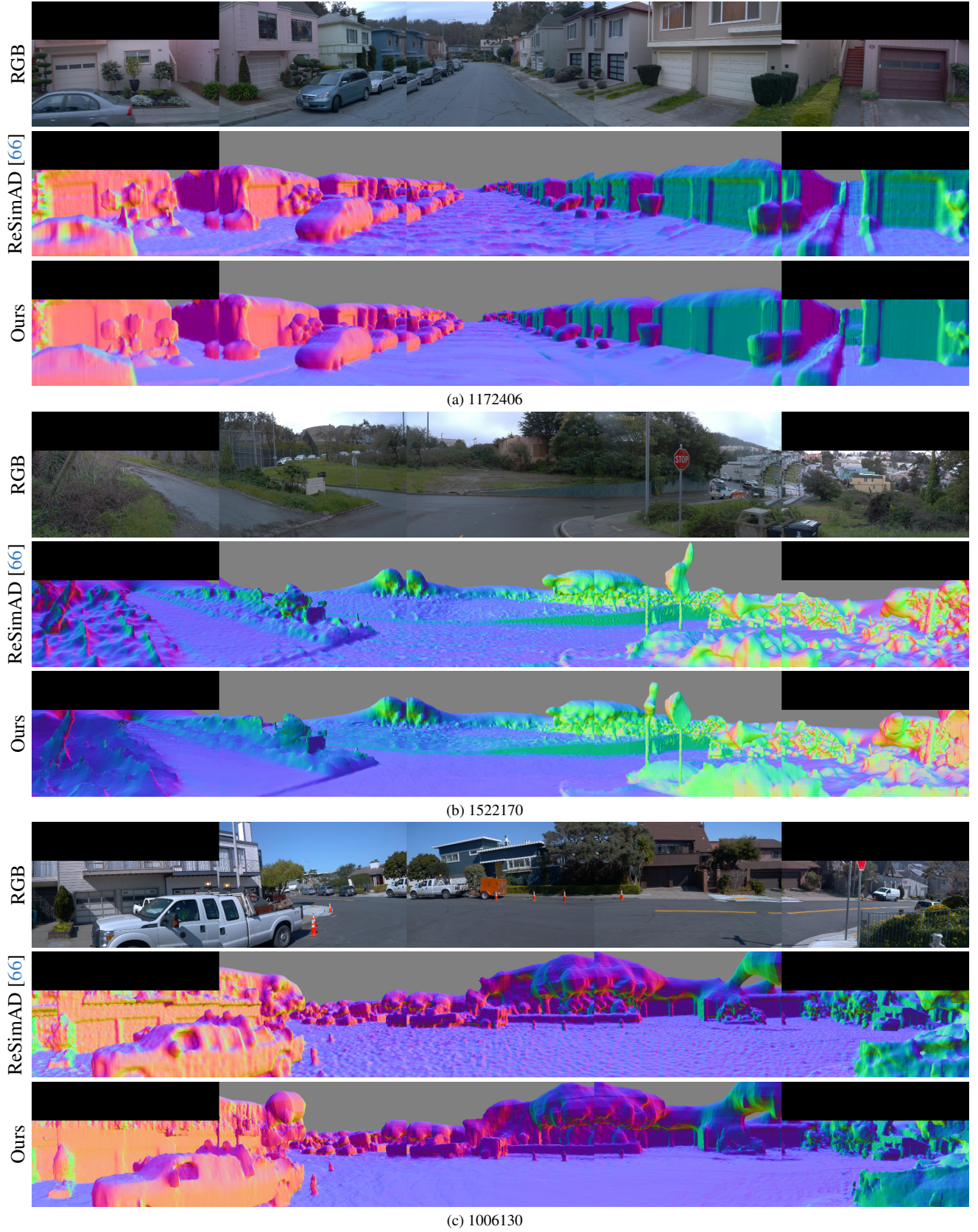
Figure 8. Mesh reconstruction results comparing plain ReSimAD [66] and our method, which incorporates surface normals estimated from a model trained on our LiSu dataset and fine-tuned on Waymo Open Dataset [48]. Results are shown for three different Waymo sequences.
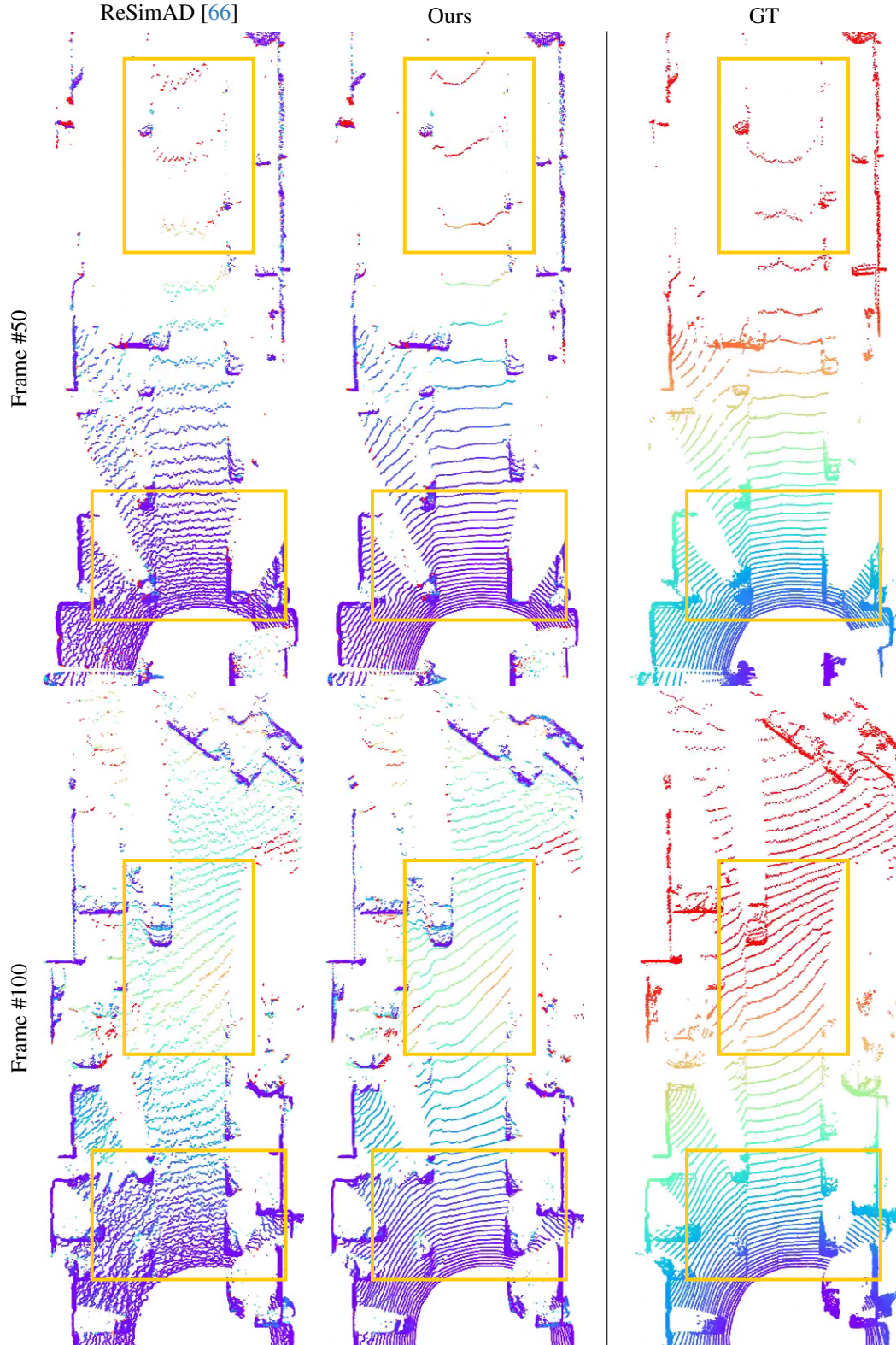
Figure 9. LiDAR simulation results comparing mesh reconstructed with plain ReSimAD [66] (left) and our method, which leverages estimated surface normals from a model trained on our LiSu dataset and fine-tuned on Waymo Open Dataset [48] (middle). The color scale represents the deviation from ground truth distance, with blue indicating low error and red indicating high error. The rightmost image shows the ground truth LiDAR point cloud (used for mesh reconstruction), colored by distance from the sensor (blue: near, red: far).

	Map	# Runs	# Frames	Summary
train	Town01	11	6339	A small, simple town with a river and several bridges.
	Town03	11	7658	A larger, urban map with a roundabout and large junctions.
	Town05	11	5477	Squared-grid town with cross junctions and a bridge. It has multiple lanes per direction.
	Town07	11	5579	A rural environment with narrow roads, corn, barns and hardly any traffic lights.
total	4	44	25 053	
test	Town02	11	5235	A small simple town with a mixture of residential and commercial buildings.
	Town04	5	3591	A small town embedded in the mountains with a special "figure of 8"' infinite highway.
	Town06	11	3980	Long many lane highways with many highway entrances and exits (with Michigan left).
	Town12	16	9361	A large map, including high-rise, residential and rural environments.
total	4	43	22 167	
val	Town10	11	2825	A downtown area with skyscrapers, residential buildings and an ocean promenade.
total	1	11	2825	
overall	9	98	50 045	

Table 6. Summary of our data splits, including CARLA [13] maps, number of randomly started simulation runs, and total number of frames for the given map. We include the short summary provided by CARLA for each map.

Seq.	ReSimAD [66]	Ours
1027514	0.68 / 0.15	**0.65 / 0.14**
1006130	**2.55 / 0.23**	2.56 / 0.25
1137922	0.95 / 0.77	**0.92 / 0.73**
1323841	**0.51 / 0.11**	**0.51** / 0.12
1486973	0.83 / **0.16**	**0.82** / 0.17
1522170	2.05 / 2.70	**1.78 / 2.06**
1647019	1.01 / 0.70	**0.98 / 0.63**
3425716	0.70 / 0.22	**0.68 / 0.20**
9385013	0.52 / 0.20	**0.51 / 0.18**
average	1.09 / 0.58	**1.04 / 0.50**

Table 7. We evaluate neural surface reconstruction on diverse Waymo sequences using Root Mean Square Error (RMSE) / Chamfer Distance (CD). Lower values indicate better performance. ReSimAD [66] omits surface normal loss during reconstruction. Ours is a Waymo model, trained with our proposed self-training framework.

E. Qualitative Evaluation

We conduct a qualitative evaluation on the Waymo Open Dataset [48] to highlight the benefits of our LiSu. By comparing SHS-Net [31] trained on PCPNet [21] to our LiSu, we demonstrate the significant advantage of leveraging a dataset tailored to real-world LiDAR point clouds. The lack of publicly available LiDAR datasets underscores the potential of our LiSu to advance the field, regardless of the specific method employed. Notably, our self-supervised approach achieves impressive results when applied to a real-world dataset like Waymo, as visualized in Fig. 10.

F. Loss-Regularization Trade-off

In the following section, we present an ablation study to substantiate our selection of the γ parameter (Eq. (7) of the main manuscript), which balances loss minimization and regularization. To expedite the training and evaluation phases, we utilized 50% and 20% subsets of the original training and evaluation data, respectively.
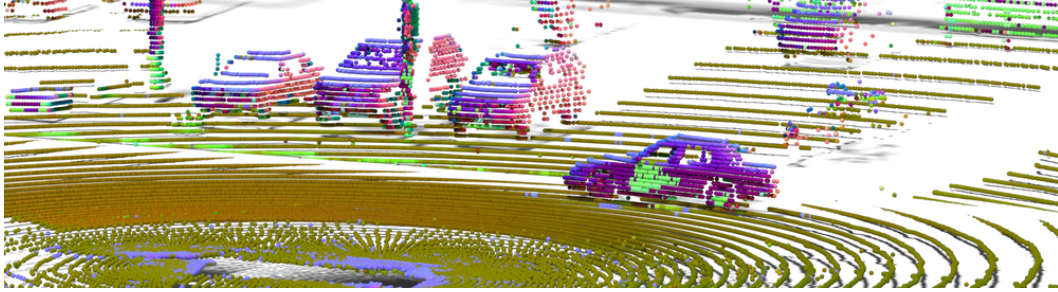
The hyperparameter γ balances the trade-off between noise and smoothness in the final predictions. Lower values of γ encourage the model to prioritize edge preservation, potentially leading to noisier predictions. Conversely, higher values of γ promote smoother predictions, which may result in the loss of fine-grained details and edge information. In the extreme case, when $\gamma = 1$, the model's predictions become almost entirely smooth, with minimal edge detection. In our experiments, we found that $\gamma = 0.1$ provided an optimal balance between noise and detail, as illustrated in Fig. 11.

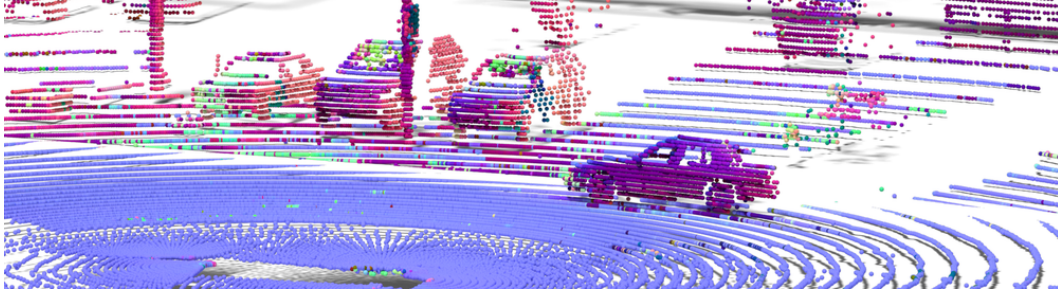## G. k Neighborhood Graph Ablation

To construct the k-neighborhood graph $\mathcal{G}$ for both Spatial Graph Total Variation (SGTV) and Temporal Graph Total Variation (TGTV) (Sec. 3.2), we require a hyperparameter k to specify the size of the local neighborhood. In the following experiments, we fix $\gamma = 0.1$ and systematically vary k to assess its influence on the model's overall performance. To expedite the training and evaluation phases, we utilized 50% and 20% subsets of the original training and evaluation data, respectively.

The parameter k controls the model's output smoothness, with higher values leading to increased smoothing and potential loss of detail (*e.g.* $k = 32$ in Fig. 12). Conversely, smaller
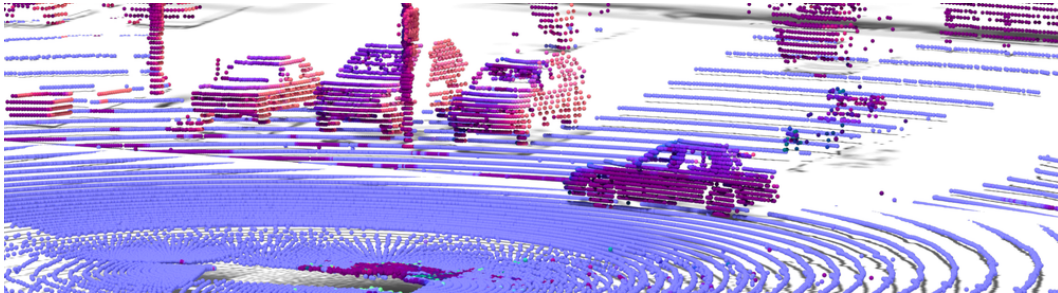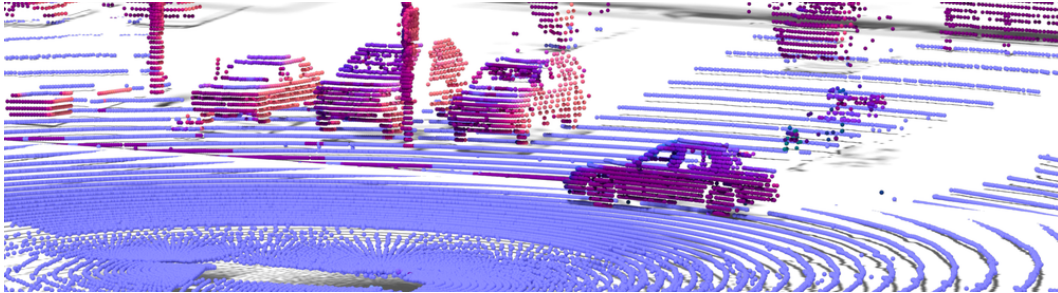
(a) SHS-Net [31] trained on PCPNet [21]

(b) SHS-Net [31] trained on our LiSu

(c) Direct transfer with our method

(d) Unsupervised domain adaptation from LiSu to Waymo with our method

Figure 10. Qualitative comparison of SHS-Net [31] directly transferred from PCPNet [21] (a) and our LiSu dataset (b). Our dataset, tailored for LiDAR surface normal estimation, yields superior results. Additionally, we demonstrate the effectiveness of our method for both direct transfer (c) and self-supervised domain adaptation (d) on a challenging Waymo Open Dataset frame.

values of k (*e.g.* 4) may not provide sufficient smoothing, resulting in noisy outputs. As Fig. 12 illustrates, $k = 8$ offers a favorable balance between smoothness and detail preservation. As a general guideline, we recommend setting k to the median number of points within a 0.1-meter radius around each point in the input data.

H. Weighted Adjacency Matrix Ablation

Edge weights in our graph $\mathcal{G}$ for both SGTV and TGTV (Sec. 3.2 of the main manuscript) are computed using an exponential decay function,

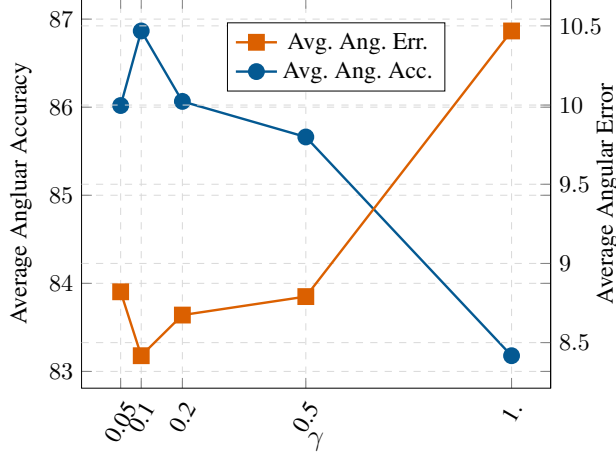$$w(x) = \exp\left(-\frac{x^2}{\sigma^2}\right), \tag{8}$$

Figure 11. Average angular accuracy (↑) computed across thresholds $\{5.0°, 7.5°, 11.25°, 22.5°, 30.0°\}$ and average angular error (↓) calculated across mean, median and RMSE, for varying γ.
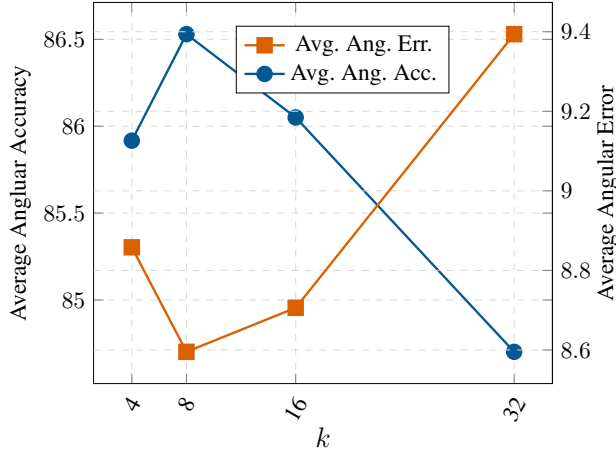

Figure 12. Average angular accuracy (↑) computed across thresholds $\{5.0°, 7.5°, 11.25°, 22.5°, 30.0°\}$ and average angular error (↓) calculated across mean, median and RMSE, for varying k.

where x represents the Euclidean distance between two graph nodes (*i.e.* points). The decay constant σ determines the rate at which the edge weight decreases with increasing distance. We depict the influence of different σ in Fig. 13.

To study the impact of the hyperparameter σ, we conducted an ablation study with $\gamma = 0.1$ and $k = 8$, varying σ across multiple runs. For efficiency, we used 50% of the training data and 20% of the evaluation data. Results showed that smaller σ values produce sparse graphs, reducing regularization, while larger values introduce noise by connecting distant points, potentially belonging to different surfaces (*e.g.* 0.2 meters apart). Empirically, $\sigma = 0.1$ was found optimal, as shown in Figure 14.
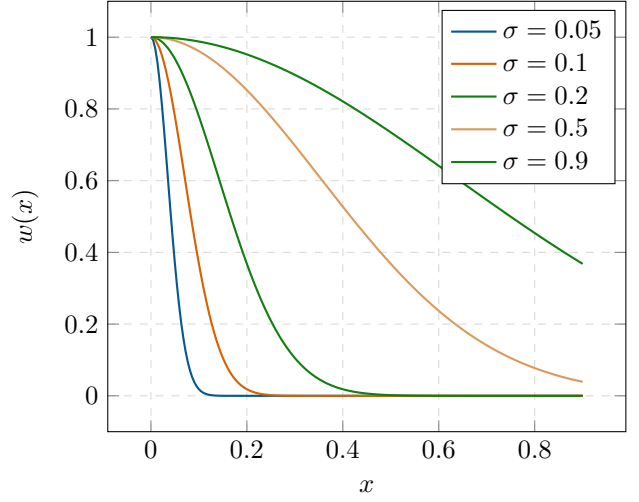

Figure 13. Effect of the decay constant σ on edge weights for different distances x in meters.
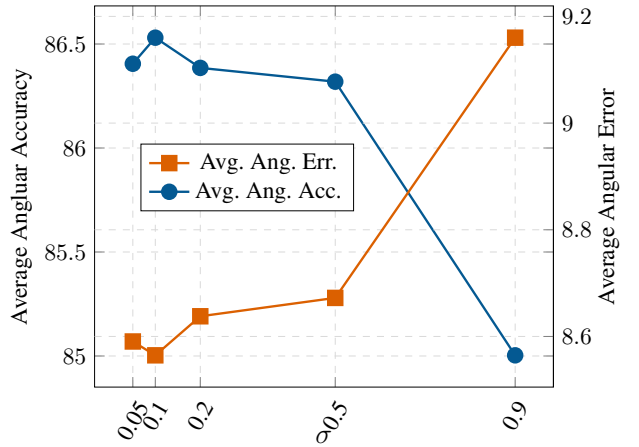

Figure 14. Average angular accuracy (↑) computed across thresholds $\{5.0°, 7.5°, 11.25°, 22.5°, 30.0°\}$ and average angular error (↓) calculated across mean, median and RMSE, for varying σ.