DiffSF: Diffusion Models for Scene Flow Estimation

Yushan Zhang **Bastian Wandt** **Maria Magnusson** **Michael Felsberg**
Linköping University
{firstname.lastname}@liu.se

Abstract

Scene flow estimation is an essential ingredient for a variety of real-world applications, especially for autonomous agents, such as self-driving cars and robots. While recent scene flow estimation approaches achieve reasonable accuracy, their applicability to real-world systems additionally benefits from a reliability measure. Aiming at improving accuracy while additionally providing an estimate for uncertainty, we propose *DiffSF* that combines transformer-based scene flow estimation with denoising diffusion models. In the diffusion process, the ground truth scene flow vector field is gradually perturbed by adding Gaussian noise. In the reverse process, starting from randomly sampled Gaussian noise, the scene flow vector field prediction is recovered by conditioning on a source and a target point cloud. We show that the diffusion process greatly increases the robustness of predictions compared to prior approaches resulting in state-of-the-art performance on standard scene flow estimation benchmarks. Moreover, by sampling multiple times with different initial states, the denoising process predicts multiple hypotheses, which enables measuring the output uncertainty, allowing our approach to detect a majority of the inaccurate predictions. The code is available at https://github.com/ZhangYushan3/DiffSF.

1 Introduction

Scene flow estimation is an important research topic in computer vision with applications in various fields, such as autonomous driving [25] and robotics [30]. Given a source and a target point cloud, the objective is to estimate a scene flow vector field that maps each point in the source point cloud to the target point cloud. Many studies on scene flow estimation aim at enhancing accuracy and substantial progress has been made particularly on clean, synthetic datasets. However, real-world data contains additional challenges such as severe occlusion and noisy input, thus requiring a high level of robustness when constructing models for scene flow estimation.

Recently, Denoising Diffusion Probabilistic Models (DDPMs) have not only been widely explored in image generation [12, 28] but also in analysis tasks, e.g. detection [3], classification [11], segmentation [1, 10], optical flow [29], human pose estimation [13], point cloud registration [14], etc. Drawing inspiration from the recent successes of diffusion models in regression tasks and recognizing their potential compatibility with scene flow estimation, we formulate scene flow estimation as a diffusion process following DDPMs [12] as shown in Figure 1. The forward process initiates from the ground truth scene flow vector field and gradually introduces noise to it. Conversely, the reverse process is conditioned on the source and the target point cloud and is tasked to reconstruct the scene flow vector field based on the current noisy input. To learn the denoising process, a new network is proposed inspired by state-of-the-art scene flow estimation methods FLOT [26] and GMSF [43].

Previous methods [43, 5, 36, 4] usually suffer from inaccuracies when occlusions occur or when dealing with noisy inputs. During inference, based on the fixed parameters learned during training, they cannot provide information about their inaccurate predictions, which might lead to problems in safety-critical downstream tasks. Our proposed method approaches this problem in two aspects:

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-3532

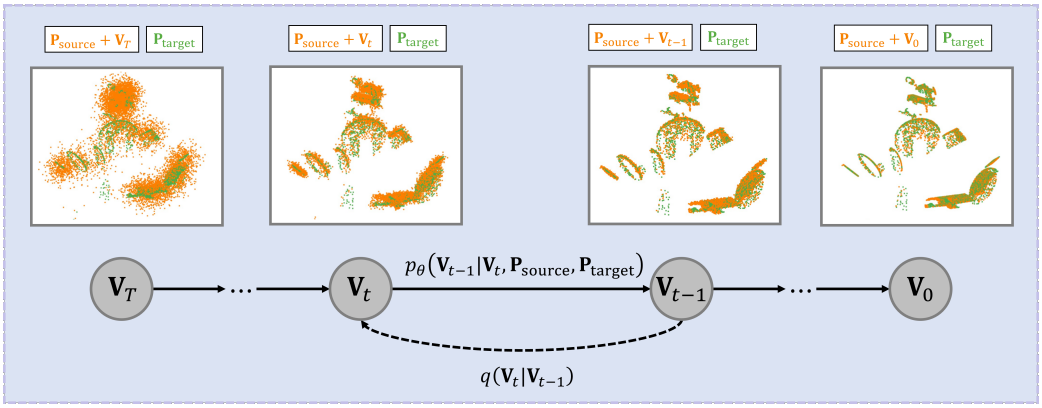

Figure 1: **Diffusion process.** In the forward process, we start from a ground truth scene flow vector field $\mathbf{V}_0$ and gradually add noise to it until we reach $\mathbf{V}_T$, which is completely Gaussian noise. In the reverse process, we recover the scene flow vector field $\mathbf{V}_0$ from the randomly sampled noisy vector field $\mathbf{V}_T$ conditioned on the source point cloud $\mathbf{P}_{\text{source}}$ and the target point cloud $\mathbf{P}_{\text{target}}$.

First, denoising diffusion models are capable of handling noisy data by modeling stochastic processes. The noise caused by sensors in the real world is filtered out, which allows the model to focus on learning underlying patterns. By learning feature representations that are robust to noise, the prediction accuracy is improved. Second, since the diffusion process introduces randomness into the inherently deterministic prediction task, it can provide a measure of uncertainty for each prediction by averaging over a set of hypotheses, notably without any modifications to the training process. Extensive experiments on multiple benchmarks, FlyingThings3D [24], KITTI Scene Flow [25], and Waymo-Open [33], demonstrate state-of-the-art performance of our proposed method. Furthermore, we demonstrate that the predicted uncertainty correlates with the prediction error, establishing it as a reasonable measure that can be adjusted to the desired certainty level with a simple threshold value.

To summarize, our contributions are: (1) We introduce DiffSF, leveraging diffusion models to solve the full scene flow estimation problem, where the inherent noisy property of the diffusion process filters out noisy data, thus, increasing the focus on learning the relevant patterns. (2) DiffSF introduces randomness to the scene flow estimation task, which allows us to predict the uncertainty of the estimates without being explicitly trained for this purpose. (3) We develop a novel architecture that combines transformers and diffusion models for the task of scene flow estimation, improving both accuracy and robustness for a variety of datasets.

2 Related Work

Scene Flow Estimation has rapidly progressed since the introduction of FlyingThings3D [24], KITTI Scene Flow [25], and Waymo-Open [33] benchmarks. Many existing methods [2, 23, 25, 27, 31, 35, 42] assume scene objects are rigid and break down the estimation task into sub-tasks involving object detection or segmentation, followed by motion model fitting. While effective for autonomous driving scenes with static background and moving vehicles, these methods struggle with more complex scenes containing deformable objects, and their non-differentiable components impede end-to-end training without instance-level supervision. Recent advancements in scene flow estimation focus on end-to-end trainable models and are categorized into encoder-decoder, coarse-to-fine, recurrent, soft correspondence methods, and runtime optimization-based methods. Encoder-decoder techniques, exemplified by FlowNet3D [22, 39] and HPLFlowNet [9], utilize neural networks to learn scene flow by adopting an hourglass architecture. Coarse-to-fine methods, such as PointPWC-Net [41], progressively estimate motion from coarse to fine scales, leveraging hierarchical feature extraction and warping. Recurrent methods like FlowStep3D [17], PV-RAFT [40], and RAFT3D [34] iteratively refine the estimated motion, thus enhancing accuracy. Some approaches like FLOT [26], STCN[18], and GMSF [43] frame scene flow estimation as an optimal transport problem, employing convolutional layers and point transformer modules for correspondence computation. Different from the previously mentioned methods, which are fully trained and supervised offline, the runtime optimization-based

methods [19, 20, 6] are optimized during the evaluation time based on each pair of inputs. While these methods have the advantage of without the need for training datasets, it also means that they can not take advantage of large-scale training datasets. Due to the online optimization, they also suffer from slow inference speed. Moreover, most of them focus only on autonomous driving scenes. On the other hand, we aim to estimate the scene flow of more general scenarios. Our proposed method takes the current state-of-the-art soft correspondence method GMSF [43] as a baseline. Given the fact that being able to indicate uncertainty of the estimation is an important feature for safety-critical downstream tasks, we propose to leverage the diffusion models for this purpose, whose ability of uncertainty indication has been proven by other relevant research areas [11, 29].

Diffusion Models for Regression. Diffusion models have been widely exploited for image generation [12, 28]. Beyond their capacity to generate realistic images and videos, researchers have also explored their potential to approach regression tasks. CARD [11] introduces a classification and regression diffusion model to accurately capture the mean and the uncertainty of the prediction. DiffusionDet [3] formulates object detection as a denoising diffusion process from noisy boxes to object boxes. Baranchuk *et al*. [1] employ diffusion models for semantic segmentation with scarce labeled data. DiffusionInst [10] depicts instances as instance-aware filters and casts instance segmentation as a denoising process from noise to filter. Jiang *et al*. [14] introduce diffusion models to point cloud registration that operates on the rigid body transformation group. Recent research on optical flow and depth estimation [29] shows the possibility of using diffusion models for dense vision tasks. While there have been attempts to employ diffusion models for scene flow estimation [21], they mainly focus on refining an initial estimation. On the contrary, our goal is to construct a model to estimate the full scene flow vector field instead of a refinement plug-in module. To the best of our knowledge, we are the first to propose using diffusion models to estimate the full scene flow directly from two point clouds.

3 Proposed Method

3.1 Preliminaries

Scene Flow Estimation. Given a source point cloud $\mathbf{P}_{\text{source}} \in \mathbb{R}^{N_1 \times 3}$ and a target point cloud $\mathbf{P}_{\text{target}} \in \mathbb{R}^{N_2 \times 3}$, where N_1 and N_2 are the number of points in the source and the target point cloud respectively, the objective is to estimate a scene flow vector field $\mathbf{V} \in \mathbb{R}^{N_1 \times 3}$ that maps each source point to the correct position in the target point cloud.

Diffusion Models. Inspired by non-equilibrium thermodynamics, diffusion models [12, 32] are a class of latent variable $(x_1, ..., x_T)$ models of the form $p_\theta(x_0) = \int p_\theta(x_{0:T}) dx_{1:T}$, where the latent variables are of the same dimensionality as the input data x_0 (any dimensionality). The joint distribution $p_\theta(x_{0:T})$ is also called the *reverse process*

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \quad p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)). \tag{1}$$

The approximate posterior $q(x_{1:T}|x_0)$ is called the *forward process*, which is fixed to a Markov chain that gradually adds noise according to a predefined noise scheduler $\beta_{1:T}$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \quad q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t \mathbf{I}). \tag{2}$$

The training is performed by minimizing a variational bound on the negative log-likelihood

$$\begin{aligned} \mathbb{E}_q[-\log p_\theta(x_0)] &\leq \mathbb{E}_q[-\log \tfrac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}] \\ &= \mathbb{E}_q[D_{\text{KL}}(q(x_T|x_0) \| p(x_T)) \\ &\quad + \textstyle\sum_{t>1} D_{\text{KL}}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)) - \log p_\theta(x_0|x_1)], \end{aligned} \tag{3}$$

where D_{KL} denotes the Kullback–Leibler divergence.

3.2 Scene Flow Estimation as Diffusion Process

We formulate the scene flow estimation task as a conditional diffusion process that is illustrated in Figure 1. The *forward process* starts from the ground truth scene flow vector field $\mathbf{V}_0$ and ends at

https://doi.org/10.52202/079017-3532

pure Gaussian noise $\mathbf{V}_T$ by gradually adding Gaussian noise to the input data as in Eq. (2). Given that β_t is small, $q(\mathbf{V}_t|\mathbf{V}_{t-1})$ in Eq. (2) has a closed form [12]

$$q(\mathbf{V}_t|\mathbf{V}_0) = \mathcal{N}(\mathbf{V}_t; \sqrt{\bar{\alpha}_t}\mathbf{V}_0, (1-\bar{\alpha}_t)\mathbf{I}), \tag{4}$$

where $\bar{\alpha}_t := \prod_{s=1}^{t}(1-\beta_s)$. The *reverse process* predicts the ground truth $\mathbf{V}_0$ from the noisy input $\mathbf{V}_t$ conditioned on both the source point cloud $\mathbf{P}_{\text{source}}$ and the target point cloud $\mathbf{P}_{\text{target}}$,

$$p_\theta(\mathbf{V}_{t-1}|\mathbf{V}_t, \mathbf{P}_{\text{source}}, \mathbf{P}_{\text{target}}) = \mathcal{N}(\mathbf{V}_{t-1}; \mu_\theta(\mathbf{V}_t, \mathbf{P}_{\text{source}}, \mathbf{P}_{\text{target}}), \mathbf{I}). \tag{5}$$

The forward process posterior is tractable when conditioned on $\mathbf{V}_0$,

$$q(\mathbf{V}_{t-1}|\mathbf{V}_t, \mathbf{V}_0) = \mathcal{N}(\mathbf{v}_{t-1}; \tilde{\mu}_t(\mathbf{V}_t, \mathbf{V}_0), \tilde{\beta}_t\mathbf{I}), \tag{6}$$

where $\tilde{\mu}_t(\mathbf{V}_t, \mathbf{V}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{V}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{V}_t$, and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$. Minimizing the variational bound in Eq. (3) breaks down to minimizing the difference between $\tilde{\mu}_t(\mathbf{V}_t, \mathbf{V}_0)$ and $\mu_\theta(\mathbf{V}_t, \mathbf{P}_{\text{source}}, \mathbf{P}_{\text{target}})$. Since $\mathbf{V}_t$ is constructed from $\mathbf{V}_0$ by a predefined fixed noise scheduler $\beta_{1:T}$, the training objective is further equivalent to learning $\mathbf{V}_0$ by a neural network $f_\theta(\mathbf{V}_t, \mathbf{P}_{\text{source}}, \mathbf{P}_{\text{target}})$. The training loss can be written as

$$\mathcal{L} = \|f_\theta(\mathbf{V}_t, \mathbf{P}_{\text{source}}, \mathbf{P}_{\text{target}}) - \mathbf{V}_0\|, \tag{7}$$

where the neural network $f_\theta(\mathbf{V}_t, \mathbf{P}_{\text{source}}, \mathbf{P}_{\text{target}})$ takes the current noisy input $\mathbf{V}_t$, the source point cloud $\mathbf{P}_{\text{source}}$, and the target point cloud $\mathbf{P}_{\text{target}}$ as input and output $\hat{\mathbf{V}}_{\text{pred}}$, which is an prediction of $\mathbf{V}_0$. The detailed architecture of f_θ is presented in section 3.3. The reverse process in Eq. (5) can be rewritten by replacing μ_θ with f_θ as

$$p_\theta(\mathbf{V}_{t-1}|\mathbf{V}_t, \mathbf{P}_{\text{source}}, \mathbf{P}_{\text{target}}) = \mathcal{N}(\mathbf{V}_{t-1}; \tilde{\mu}_t(\mathbf{V}_t, f_\theta(\mathbf{V}_t, \mathbf{P}_{\text{source}}, \mathbf{P}_{\text{target}})), \mathbf{I}). \tag{8}$$

During *inference*, starting from randomly sampled Gaussian noise $\mathbf{V}_T$, $\mathbf{V}_0$ is reconstructed with the model f_θ according to the reverse process in Eq. (8). The detailed training and sampling algorithms are given in Algorithm 1 and Algorithm 2.

Algorithm 1: Training

repeat
  $\mathbf{V}_0 \sim q(\mathbf{V}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $t \sim$ Uniform($\{1, ..., T\}$);
  $\mathbf{V}_t = \sqrt{\bar{\alpha}_t}\mathbf{V}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$;
  estimate $\hat{\mathbf{V}}_{\text{pred}} = f_\theta(\mathbf{V}_t, \mathbf{P}_{\text{source}}, \mathbf{P}_{\text{target}})$;
  optimize loss: $\mathcal{L}_t = \text{loss}(\hat{\mathbf{V}}_{\text{pred}}, \mathbf{V}_0)$;
until *converged*;

Algorithm 2: Sampling

$\mathbf{V}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
for $t = T, ..., 1$ **do**
  estimate $\hat{\mathbf{V}}_{\text{pred}} = f_\theta(\mathbf{V}_t, \mathbf{P}_{\text{source}}, \mathbf{P}_{\text{target}})$;
  if $t > 1$: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
  else: $\mathbf{z} = \mathbf{0}$;
  $\mathbf{V}_{t-1} = \tilde{\mu}_t(\mathbf{V}_t, \hat{\mathbf{V}}_{\text{pred}}) + \mathbf{z}$;
return $\mathbf{V}_0$;

3.3 Architecture

To train the diffusion process with Eq. (7), we need to design the neural network to predict $\mathbf{V}_0$, i.e. the ground truth scene flow vector field. The reverse process with the detailed architecture of $\hat{\mathbf{V}}_{\text{pred}} = f_\theta(\mathbf{V}_t, \mathbf{P}_{\text{source}}, \mathbf{P}_{\text{target}})$ is given in Figure 2. We take the state-of-the-art method GMSF [43] as our baseline. All the building blocks, Feature Extraction, Local-Global-Cross Transformer, and Global Correlation are the same as in GMSF [43]. We modify the model architecture of GMSF following the recent work [26, 8, 17] of scene flow estimation by adding an initial estimation before the final prediction. More specifically, the source point cloud $\mathbf{P}_{\text{source}} \in \mathbb{R}^{N_1 \times 3}$ is first warped with $\mathbf{V}_t \in \mathbb{R}^{N_1 \times 3}$. The warped source point cloud and the target point cloud are sent to the Feature Extraction block to expand the three-dimensional coordinate into higher-dimensional features for each point. Based on the similarities between point pairs in the warped source and the target point cloud, a Global Correlation is applied to compute an initial estimation $\hat{\mathbf{V}}_{\text{init}} \in \mathbb{R}^{N_1 \times 3}$. We then warp the source point cloud $\mathbf{P}_{\text{source}} \in \mathbb{R}^{N_1 \times 3}$ with the initial estimation $\hat{\mathbf{V}}_{\text{init}} \in \mathbb{R}^{N_1 \times 3}$. The same Feature Extraction block is applied on both the warped source point cloud and the target point cloud, but with different weights than the previous block. A Local-Global-Cross Transformer is then applied to the higher-dimensional features to get a more robust and reliable feature representation for each point. The output features are then sent into the Global Correlation block to get the final prediction $\hat{\mathbf{V}}_{\text{pred}} \in \mathbb{R}^{N_1 \times 3}$. The detailed architecture of Feature Extraction, Local-Global-Cross Transformer, and Global Correlation is given in the following paragraphs using the same notation as GMSF [43].

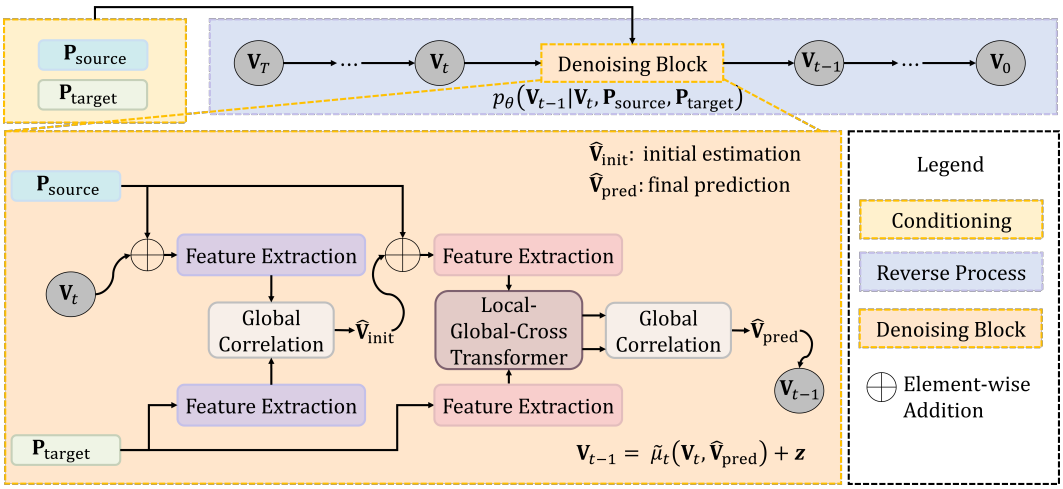

Figure 2: The reverse process with detailed denoising block for scene flow estimation. The denoising block takes the current noisy input $\mathbf{V}_t$, the source point cloud $\mathbf{P}_{\text{source}}$, and the target point cloud $\mathbf{P}_{\text{target}}$ as input. The output $\hat{\mathbf{V}}_{\text{pred}}$ is the denoised scene flow prediction. Shared weights for the feature extraction are indicated in the same color.

Feature Extraction The three-dimensional coordinate for each point is first projected into a higher feature dimension $\mathbf{x}_i^h \in \mathbb{R}^{1\times d}$ by the off-the-shelf feature extraction backbone DGCNN [38]. Each layer of the network can be written as

$$\mathbf{x}_i^h = \max_{\mathbf{x}_j \in \mathcal{N}(i)} h(\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i), \tag{9}$$

where i and j denote the index of a single point in the point cloud. $\mathbf{x}_j \in \mathcal{N}(i)$ denotes the neighboring points of point $\mathbf{x}_i$ found by a k-nearest-neighbor (KNN) algorithm. The number of k is set to 16. The point feature $\mathbf{x}_i$ and the edge feature $\mathbf{x}_j - \mathbf{x}_i$ are first concatenated together along the feature dimension and then passed through a neural network h. h consists of a sequence of linear layer, batch normalization, and leaky ReLU layer. The output feature dimension d is set to 128. The maximum value of the k nearest neighbors is taken as the output. Multiple layers are stacked together to get the final feature representation $\mathbf{x}_i^h$.

Local-Global-Cross Transformer takes the output high-dimensional features $\mathbf{x}_i^h \in \mathbb{R}^{1\times d}$ as input to learn more robust and reliable feature representations,

$$\mathbf{x}_i^l = \textstyle\sum_{\mathbf{x}_j \in \mathcal{N}(i)} \gamma(\varphi_l(\mathbf{x}_i^h) - \psi_l(\mathbf{x}_j^h) + \delta) \odot (\alpha_l(\mathbf{x}_j^h) + \delta), \tag{10}$$

$$\mathbf{x}_i^g = \textstyle\sum_{\mathbf{x}_j \in \mathcal{X}_1} \langle \varphi_g(\mathbf{x}_i^l), \psi_g(\mathbf{x}_j^l) \rangle \alpha_g(\mathbf{x}_j^l), \tag{11}$$

$$\mathbf{x}_i^c = \textstyle\sum_{\mathbf{x}_j \in \mathcal{X}_2} \langle \varphi_c(\mathbf{x}_i^g), \psi_c(\mathbf{x}_j^g) \rangle \alpha_c(\mathbf{x}_j^g), \tag{12}$$

where local, global, and cross transformers are given in Eq. (10) (11) (12) respectively. φ, ψ, and α denote linear layers to generate the query, key, and value. The indices $\cdot_l$, $\cdot_g$, and $\cdot_c$ indicate local transformer, global transformer, and cross transformer, respectively. For the local transformer, γ is a sequence of linear layer, ReLU, linear layer, and softmax. δ is the relative positional embedding that gives the information of the 3D coordinate distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. $\odot$ denotes element-wise multiplication. The output $\mathbf{x}_i^l$ is further processed by a linear layer and a residual connection from the input before being sent to the global transformer. For the global and cross transformer, $\mathcal{X}_1 = \mathbf{P}_{\text{source}} + (\mathbf{V}_t \text{ or } \hat{\mathbf{V}}_{\text{init}}) \in \mathbb{R}^{N_1\times 3}$ and $\mathcal{X}_2 = \mathbf{P}_{\text{target}} \in \mathbb{R}^{N_2\times 3}$ represent the warped source point cloud and the target point cloud, respectively. $\langle , \rangle$ denotes the scalar product. The output of the global and cross transformer is further processed by a linear layer, a layer normalization, and a residual connection from the input. A feedforward network with a multilayer perceptron and layer normalization is applied to the output of the cross transformer to aggregate information. To acquire more robust feature representations, the global-cross transformers are stacked and repeated multiple times (14 times in our experiment). For simplicity, we only give the equations for learning the features of $\mathcal{X}_1$. The features of $\mathcal{X}_2$ are computed by the same procedure. The output point features $\mathbf{x}_i^c$ and $\mathbf{x}_j^c$ for each point cloud are stacked together to form feature matrices $\mathbf{F}_1 \in \mathbb{R}^{N_1\times d}$ and $\mathbf{F}_2 \in \mathbb{R}^{N_2\times d}$.

https://doi.org/10.52202/079017-3532

Global Correlation predicts the scene flow vector solely based on two feature similarity matrices, cross feature similarity matrix $\mathbf{M}_{\text{cross}} \in \mathbb{R}^{N_1 \times N_2}$ and self feature similarity matrix $\mathbf{M}_{\text{self}} \in \mathbb{R}^{N_1 \times N_1}$.

$$\mathbf{M}_{\text{cross}} = \text{softmax}(\mathbf{F}_1 \mathbf{F}_2^T / \sqrt{d}), \tag{13}$$

$$\mathbf{M}_{\text{self}} = \text{softmax}(W_q(\mathbf{F}_1) W_k(\mathbf{F}_1)^T / \sqrt{d}), \tag{14}$$

where W_q and W_k are linear projections. d is the feature dimensions. The softmax is taken over the second dimension of the matrices. The cross feature similarity matrix $\mathbf{M}_{\text{cross}} \in \mathbb{R}^{N_1 \times N_2}$ encodes the feature similarities between all the points in the source point cloud $\mathbf{P}_{\text{source}}$ and all the points in the target point cloud $\mathbf{P}_{\text{target}}$. The self feature similarity matrix $\mathbf{M}_{\text{self}} \in \mathbb{R}^{N_1 \times N_1}$ encodes the feature similarities between all points in the source point cloud $\mathbf{P}_{\text{source}}$. The global correlation is performed by a matching process guided by the cross feature similarity matrix followed by a smoothing procedure guided by the self feature similarity matrix

$$\hat{\mathbf{V}} = \mathbf{M}_{\text{self}}(\mathbf{M}_{\text{cross}} \mathbf{P}_{\text{target}} - \mathbf{P}_{\text{source}}). \tag{15}$$

We follow GMSF [43] and employ a robust loss defined as

$$\mathcal{L} = \textstyle\sum_i (\|\hat{\mathbf{V}}_{\text{pred}}(i) - \mathbf{V}_{\text{gt}}(i)\|_1 + \epsilon)^q, \tag{16}$$

where $\hat{\mathbf{V}}_{\text{pred}}$ is the output prediction of the neural network, i.e. $f_\theta(\mathbf{V}_t, \mathbf{P}_{\text{source}}, \mathbf{P}_{\text{target}})$ in Eq. (7). $\mathbf{V}_{\text{gt}}$ denotes the ground truth scene flow vector field i.e. $\mathbf{V}_0$ in Eq. (7). i is the index of the points. ϵ is set to 0.01 and q is set to 0.4.

4 Experiments

4.1 Implementation Details

We use the AdamW optimizer and a weight decay of 1×10^{-4}. The initial learning rate is set to 4×10^{-4} for FlyingThings3D [24] and 1×10^{-4} for Waymo-Open [33]. We employ learning rate annealing by using the Pytorch OneCycleLR learning rate scheduler. During training, we set N_1 and N_2 to 4096, randomly sampled by furthest point sampling. The model is trained for 600k iterations with a batch size of 24. During inference, we follow previous methods [43, 21, 5] and set N_1 and N_2 to 8192 for a fair comparison. The number of diffusion steps is set to 20 during training and 2 during inference. The number of nearest neighbors k in DGCNN and Local Transformer is set to 16. The number of global-cross transformer layers is set to 14. The number of feature channels is set to 128. Further implementation details are given in the supplemental document and the provided code.

4.2 Evaluation Metrics

We follow the most recent work in the field [43, 21, 5] and use established evaluation metrics for scene flow estimation. EPE_{3D} measures the endpoint error between the prediction and the ground truth $\|\hat{\mathbf{V}}_{\text{pred}} - \mathbf{V}_{\text{gt}}\|_2$ averaged over all points. ACC_{S} measures the percentage of points with an endpoint error smaller than 5 cm or relative error less than 5%. ACC_{R} measures the percentage of points with an endpoint error smaller than 10 cm or relative error less than 10%. Outliers measures the percentage of points with an endpoint error larger than 30 cm or relative error larger than 10%.

4.3 Datasets

We follow the most recent work in the field [43, 21, 5] and test the proposed method on three established benchmarks for scene flow estimation.

FlyingThings3D [24] is a synthetic dataset consisting of 25000 scenes with ground truth annotations. We follow Liu *et al*. in FlowNet3D [22] and Gu *et al*. in HPLFlowNet [9] to preprocess the dataset and denote them as F3D_{o}, with occlusions, and F3D_{s}, without occlusions. The former consists of 20000 and 2000 scenes for training and testing, respectively. The latter consists of 19640 and 3824 scenes for training and testing, respectively.

KITTI Scene Flow [25] is a real autonomous driving dataset with 200 scenes for training and 200 scenes for testing. Since the annotated data in KITTI is limited, the dataset is mainly used

for evaluating the generalization ability of the models trained on FlyingThings3D. Similar to the FlyingThings3D dataset, following Liu *et al*. in FlowNet3D [22] and Gu *et al*. in HPLFlowNet [9], the KITTI dataset is preprocessed as KITTI_o, with occlusions, and KITTI_s, without occlusions. The former consists of 150 scenes from the annotated training set. The latter consists of 142 scenes from the annotated training set.

Waymo-Open [33] is a larger autonomous driving dataset with challenging scenes. The annotations are generated from corresponding tracked 3D objects to scale up the dataset for scene flow estimation by approximately 1000 times compared to previous real-world scene flow estimation datasets. The dataset consists of 798 training sequences and 202 testing sequences. Each sequence consists of around 200 scenes. Different preprocessing of the dataset exists [7, 15, 16], we follow the one employed in our baseline method [7].

4.4 State-of-the-art Comparison

We give state-of-the-art comparisons on multiple standard scene flow datasets. Table 1 and Table 2 show the results on the F3D_s and the F3D_o datasets, with generalization results on the KITTI_s and the KITTI_o datasets. Table 3 shows the results on the Waymo-Open dataset. On the F3D_s dataset, DiffSF shows an improvement (over the failure cases) of 31% in EPE_3D, 44% in ACC_S, 35% in ACC_R, and 45% in Outliers compared to the current state-of-the-art method GMSF [43]. Similar improvement is also shown on the F3D_o dataset with an improvement of 32% in EPE_3D, 34% in ACC_S, 24% in ACC_R, and 38% in Outliers, demonstrating DiffSF's ability to handle occlusions. The generalization abilities on the KITTI_s and the KITTI_o datasets are comparable to state of the art. All the four metrics show the best or second-best performances. On the Waymo-Open dataset, a steady improvement in both accuracy and robustness is achieved, demonstrating DiffSF's effectiveness on real-world data.

Table 1: **State-of-the-art comparison on F3D_s and KITTI_s.** The models are only trained on F3D_s without occlusions. The number of time steps is set to 20 for training and 2 for inference. The bold and the underlined numbers represent the best and the second best performance respectively.

Method	F3D_s				KITTI_s			
	$\text{EPE}_\text{3D}\downarrow$	$\text{ACC}_\text{S}\uparrow$	$\text{ACC}_\text{R}\uparrow$	Outliers $\downarrow$	$\text{EPE}_\text{3D}\downarrow$	$\text{ACC}_\text{S}\uparrow$	$\text{ACC}_\text{R}\uparrow$	Outliers $\downarrow$
FlowNet3D [22]CVPR'19	0.1136	41.25	77.06	60.16	0.1767	37.38	66.77	52.71
HPLFlowNet [9]CVPR'19	0.0804	61.44	85.55	42.87	0.1169	47.83	77.76	41.03
PointPWC [41]ECCV'20	0.0588	73.79	92.76	34.24	0.0694	72.81	88.84	26.48
FLOT [26]ECCV'20	0.0520	73.20	92.70	35.70	0.0560	75.50	90.80	24.20
Bi-PointFlow [4]ECCV'22	0.0280	91.80	97.80	14.30	0.0300	92.00	96.00	14.10
3DFlow [36]ECCV'22	0.0281	92.90	98.17	14.58	0.0309	90.47	95.80	16.12
MSBRN [5]ICCV'23	0.0150	97.30	99.20	5.60	0.0110	97.10	98.90	8.50
DifFlow3D [21]CVPR'24	0.0140	97.76	99.33	4.79	**0.0089**	98.13	99.30	**8.25**
GMSF [43]NIPS'23	0.0090	99.18	99.69	2.55	0.0215	96.22	98.25	9.84
DiffSF(ours)	**0.0062**	**99.54**	**99.80**	**1.41**	0.0098	**98.59**	**99.44**	8.31

Table 2: **State-of-the-art comparison on F3D_o and KITTI_o.** The models are only trained on F3D_o with occlusions. The number of time steps is set to 20 for training and 2 for inference.

Method	F3D_o				KITTI_o			
	$\text{EPE}_\text{3D}\downarrow$	$\text{ACC}_\text{S}\uparrow$	$\text{ACC}_\text{R}\uparrow$	Outliers $\downarrow$	$\text{EPE}_\text{3D}\downarrow$	$\text{ACC}_\text{S}\uparrow$	$\text{ACC}_\text{R}\uparrow$	Outliers $\downarrow$
FlowNet3D [22]CVPR'19	0.157	22.8	58.2	80.4	0.183	9.8	39.4	79.9
HPLFlowNet [9]CVPR'19	0.168	26.2	57.4	81.2	0.343	10.3	38.6	81.4
PointPWC [41]ECCV'20	0.155	41.6	69.9	63.8	0.118	40.3	75.7	49.6
FLOT [26]ECCV'20	0.153	39.6	66.0	66.2	0.130	27.8	66.7	52.9
Bi-PointFlow [4]ECCV'22	0.073	79.1	89.6	27.4	0.065	76.9	90.6	26.4
3DFlow [36]ECCV'22	0.063	79.1	90.9	27.9	0.073	81.9	89.0	26.1
MSBRN [5]ICCV'23	0.053	83.6	92.6	23.1	0.044	87.3	95.0	20.8
DifFlow3D [21]CVPR'24	0.047	88.2	94.0	15.0	**0.029**	**95.9**	**97.5**	**10.8**
GMSF [43]NIPS'23	0.022	95.0	97.5	5.6	0.033	91.6	95.9	13.7
DiffSF(ours)	**0.015**	**96.7**	**98.1**	**3.5**	**0.029**	94.5	97.00	13.0

4.5 Uncertainty-error Correspondence

One of the key advantages of our proposed method DiffSF compared to other approaches is that DiffSF can model uncertainty during inference, without being explicitly trained for this purpose.

https://doi.org/10.52202/079017-3532

Table 3: **State-of-the-art comparison on Waymo-Open dataset.** The number of time steps is set to 20 for training and 2 for inference.

Method	EPE_{3D} ↓	ACC_S ↑	ACC_R ↑	Outliers ↓
FlowNet3D [22]CVPR'19	0.225	23.0	48.6	77.9
PointPWC [41]ECCV'20	0.307	10.3	23.1	78.6
FESTA [37]CVPR'21	0.223	24.5	27.2	76.5
FH-Net [7]ECCV'22	0.175	35.8	67.4	60.3
GMSF [43]NIPS'23	0.083	74.7	85.1	43.5
DiffSF(ours)	**0.080**	**76.0**	**85.6**	**41.9**

With uncertainty, we refer to the epistemic uncertainty, which reflects the confidence the model has in its predictions. In our case, we predict an uncertainty for the prediction of each point. We exploit the property of diffusion models to inject randomness into inherently deterministic tasks. Without having to train multiple models, we predict multiple hypotheses using a single model with different initial randomly sampled noise.

Figure 3 shows that the standard deviation of 20 hypotheses for each point gives a reliable uncertainty estimation, which correlates very well with the inaccuracy of the prediction. Figure 3 (left) shows the relationship between the EPE and the standard deviation of the predictions averaged over the $F3D_o$ dataset. There is an almost linear correlation of the predicted uncertainty with the EPE underlining the usefulness of our uncertainty measure. Figure 3 (right) shows the recall and precision of the outlier prediction by the uncertainty. An outlier is defined as a point that has an EPE larger than 0.30 meters. The horizontal axis is the threshold applied to the uncertainty to determine the outliers. The recall is defined as the number of correctly retrieved outliers divided by the number of all the outliers. The precision is defined as the number of correctly retrieved outliers divided by the number of all the retrieved outliers. The precision-recall break-even point obtains around 55% of recall and 55% of precision.

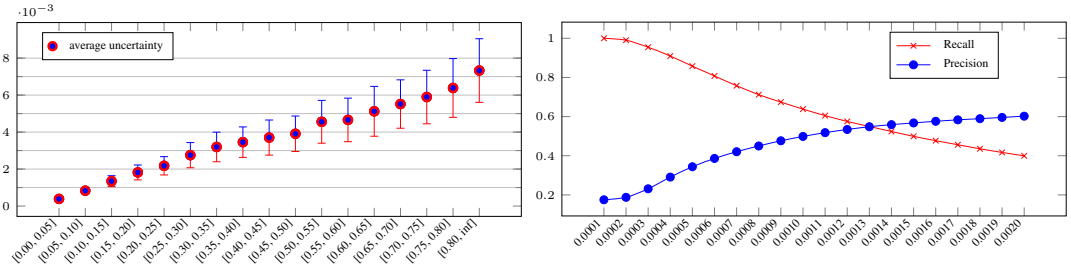

Figure 3: Analysis of uncertainty estimation on $F3D_o$ dataset. **Left**: Uncertainty-error correspondences. The horizontal axis is an interval of EPE. The vertical axis is the estimated uncertainty averaged over all the points that fall in the interval and the indication of the scaled uncertainty standard deviation. **Right**: Recall (red) and precision curve (blue) of outliers prediction. The horizontal axis is the threshold of the estimated uncertainty to determine the outliers.

Figure 4 shows visual examples that compare our outlier prediction with the actual outliers. The first row marks the scene flow estimation outliers with an EPE larger than 0.30 meters in red. The second row marks the outliers predicted by the uncertainty estimation in red. In summary, while every learned scene flow prediction model inevitably makes mistakes, our novel formulation of the task as a diffusion process not only produces state-of-the-art results but also allows for an accurate prediction of these errors. Moreover, our analysis shows that downstream tasks can select a threshold according to its desired precision and recall, therefore, mitigating potential negative effects that uncertain predictions might produce.

4.6 Ablation Study

We investigate several key design choices of the proposed method. For the denoising model architecture, we investigate how the number of global-cross transformer layers and the number of feature

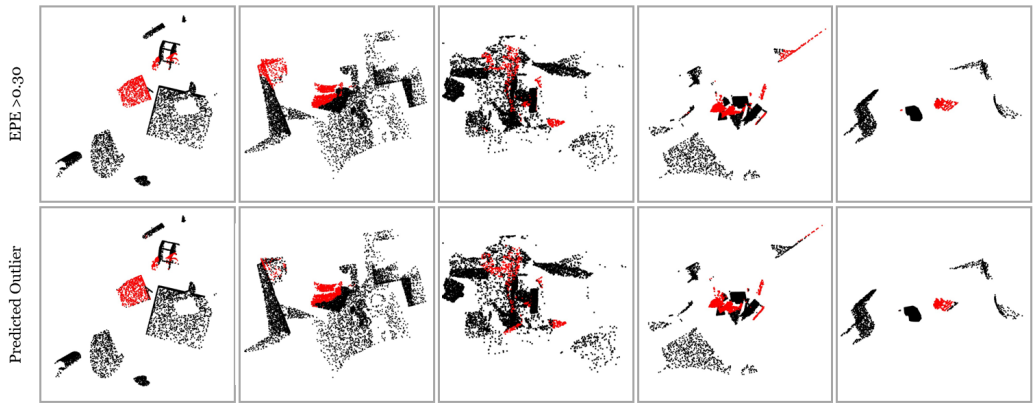

Figure 4: Visualization of outlier prediction on $F3D_o$ dataset. **Black**: Accurate prediction. **Red**: Outliers. Top row: Outliers defined as EPE > 0.30. Bottom row: Outliers predicted by Uncertainty.

channels affect the results. For the diffusion process, we investigate the influence of the number of time steps for training and sampling.

Model Architecture. To evaluate different architectural choices we select a diffusion model with five denoising blocks during training and one denoising step during testing with the DDIM [32] sampling strategy. Table 4 shows the influence of the number of global-cross transformer layers on the results. The experiments show that the best performance is achieved at the number of 14 layers. Table 5 shows the influence of the number of feature channels on the results. The experiments show that a smaller number of feature channels results in worse performance. The best performance is achieved at 128 feature channels.

Table 4: Ablation study on the number of global-cross transformer layers on $F3D_o$. The number of feature channels is set to 128. The number of time steps is set to 5 for training and 1 for inference.

Layers	EPE_{3D} ↓	ACC_S ↑	ACC_R ↑	Outliers ↓	EPE_{3D} ↓	ACC_S ↑	ACC_R ↑	Outliers ↓
	all				non-occ			
8	0.0439	91.6	94.8	7.9	0.0205	95.2	97.5	5.1
10	0.0413	92.6	95.1	7.1	0.0189	95.8	97.6	4.5
12	0.0381	93.0	95.5	6.4	0.0168	96.1	97.8	3.9
14	**0.0361**	**93.7**	**95.7**	**5.9**	**0.0153**	**96.5**	**98.0**	**3.5**
16	0.0383	93.0	95.5	6.5	0.0168	96.1	97.8	4.0

Table 5: Ablation study on the number of feature channels on $F3D_o$. The number of global-cross transformer layers is set to 14. The number of time steps is set to 5 for training and 1 for inference.

Channels	EPE_{3D} ↓	ACC_S ↑	ACC_R ↑	Outliers ↓	EPE_{3D} ↓	ACC_S ↑	ACC_R ↑	Outliers ↓
	all				non-occ			
32	0.0612	88.2	92.9	11.7	0.0299	92.9	96.3	8.2
64	0.0431	92.3	95.0	7.4	0.0199	95.7	97.5	4.7
128	**0.0361**	**93.7**	**95.7**	**5.9**	**0.0153**	**96.5**	**98.0**	**3.5**

Number of Time Steps. We set the number of global-cross transformer layers to 14 and the number of feature channels to 128. We investigate the influence of different number of time steps during training and sampling on the results. The number of time steps investigated is 5, 20, and 100 for training and 1, 2, 5, and 20 for sampling. The fast sampling is done by DDIM [32] instead of DDPM [12] sampling. Table 6 shows the results on the $F3D_o$ dataset, where $a@b$ denotes using b training steps and a sampling steps. While the results are very stable across a wide range of values, the best performance is achieved at 2@20 time steps. We hypothesize that compared to the standard setting of image generation, the lower dimensionality and variance of the scene flow data results in a smaller number of required time steps. For the number of time steps during inference, DDIM sampling works well with the best performance achieved at 2 steps.

https://doi.org/10.52202/079017-3532

Table 6: Ablation study on the number time steps for training and sampling on $F3D_o$. The number of global-cross transformer layers is set to 14. The number of feature channels is set to 128. $a@b$ denotes an inference of b training steps and a sampling steps.

Steps	EPE_{3D}(cm) ↓	ACC_S ↑	ACC_R ↑	Outliers ↓	EPE_{3D}(cm) ↓	ACC_S ↑	ACC_R ↑	Outliers ↓
	all				non-occ			
1@5	3.608	93.701	95.732	5.904	1.527	96.549	97.973	3.527
2@5	3.590	93.718	95.727	5.910	1.518	96.558	97.957	3.544
5@5	3.592	93.716	95.720	5.911	1.521	96.556	97.953	3.545
1@20	3.588	93.870	95.912	5.798	1.504	96.731	98.080	3.520
2@20	**3.576**	**93.871**	**95.919**	5.791	**1.491**	**96.736**	**98.083**	3.511
5@20	3.580	93.865	95.917	5.791	1.492	96.730	**98.083**	**3.507**
20@20	3.579	93.865	95.915	**5.789**	**1.491**	96.731	98.082	3.508
1@100	3.678	93.503	95.665	6.016	1.587	96.376	97.844	3.689
2@100	3.663	93.545	95.662	6.010	1.579	96.398	97.838	3.697
5@100	3.668	93.546	95.663	6.010	1.583	96.400	97.842	3.695
20@100	3.670	93.545	95.663	6.015	1.584	96.396	97.843	3.700

Ablation study compare to baseline GMSF. To show the improvement of our method compared to the baseline GMSF [43], we provide an additional ablation study on $F3D_o$. Since the original paper GMSF has a different training setting as our proposed DiffSF, for a fair comparison we retrain the GMSF baseline with our training setting. The result is given in Table 7 (first line). The check in the two columns denotes the implementation of improved architecture and diffusion process, respectively. The results clearly show that the proposed method DiffSF achieves superior performance than GMSF. Both the improvement of the architecture and the introduction of the diffusion process contribute to the superior performance. The improved percentage (for the introduction of the diffusion process) over the failure case is marked in the table. The results show that the proposed method has a moderate improvement in the accuracy metric EPE_{3D} and a huge improvement (more than 10%) in the robustness metrics ACC_S, ACC_R, and Outliers. Besides the better performance, the proposed method can also provide a per-prediction uncertainty.

Table 7: Ablation Study compare to baseline GMSF on $F3D_o$.

improved architecture	diffusion process	$F3D_o$-all EPE_{3D} ↓	ACC_S ↑	ACC_R ↑	Outliers ↓	$F3D_o$-nonoccluded EPE_{3D} ↓	ACC_S ↑	ACC_R ↑	Outliers ↓
		0.039	92.9	95.4	6.7	0.017	96.0	97.8	4.2
	✓	0.061	84.8	92.3	16.7	0.037	88.9	95.3	13.9
✓		0.037	93.2	95.4	6.5	0.016	96.2	97.7	4.1
✓	✓	0.036(-2.7%)	93.9(+10.3%)	95.9(+10.9%)	5.8(-10.8%)	0.015(-6.3%)	96.7(+13.2%)	98.1(+17.4%)	3.5(-14.6%)

5 Conclusions

We propose to estimate scene flow from point clouds using diffusion models in combination with transformers. Our novel approach provides significant improvements over the state-of-the-art in terms of both accuracy and robustness. Extensive experiments on multiple scene flow estimation benchmarks demonstrate the ability of DiffSF to handle both occlusions and real-world data. Furthermore, we propose to estimate uncertainty based on the randomness inherent in the diffusion process, which helps to indicate reliability for safety-critical downstream tasks. The proposed uncertainty estimation will enable mechanisms to mitigate the negative effects of potential failures.

Limitations. The training process of the diffusion models relies on annotated scene flow ground truth which is not easy to obtain for real-world data. Incorporating self-supervised training methods to leverage unannotated data might further improve our approach in the future. Furthermore, the transformer-based architecture and the global matching process limit the maximum number of points, and further research is required for peforming matching at scale.

Potential Negative Social Impact. As any other tracking algorithm, scene flow estimation can be used in surveillance scenarios, which might raise privacy concerns and ethical issues. From an ecological perspective, training of deep learning models usually takes time and resources, thus environmental impact should be taken into consideration when training and applying such compute-intensive models. However, future development in more efficient implementations will enable the positive impact of our work in e.g. robotics and autonomous driving without a significant negative impact on the environment.

Acknowledgements. This work was partly supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP), funded by Knut and Alice Wallenberg Foundation, and the Swedish Research Council grant 2022-04266; and by the strategic research environment ELLIIT funded by the Swedish government. The computational resources were provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at C3SE partially funded by the Swedish Research Council grant 2022-06725, and by the Berzelius resource, provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre.

References

[1] Dmitry Baranchuk, Andrey Voynov, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. In *International Conference on Learning Representations*, 2021.

[2] Aseem Behl, Omid Hosseini Jafari, Siva Karthik Mustikovela, Hassan Abu Alhaija, Carsten Rother, and Andreas Geiger. Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2574–2583, 2017.

[3] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19830–19843, 2023.

[4] Wencan Cheng and Jong Hwan Ko. Bi-pointflownet: Bidirectional learning for point cloud based scene flow estimation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, pages 108–124. Springer, 2022.

[5] Wencan Cheng and Jong Hwan Ko. Multi-scale bidirectional recurrent network with hybrid correlation for point cloud based scene flow estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10041–10050, 2023.

[6] Nathaniel Chodosh, Deva Ramanan, and Simon Lucey. Re-evaluating lidar scene flow for autonomous driving. *arXiv preprint arXiv:2304.02150*, 2023.

[7] Lihe Ding, Shaocong Dong, Tingfa Xu, Xinli Xu, Jie Wang, and Jianan Li. Fh-net: A fast hierarchical network for scene flow estimation on real-world point clouds. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*, pages 213–229. Springer, 2022.

[8] Xiaodong Gu, Chengzhou Tang, Weihao Yuan, Zuozhuo Dai, Siyu Zhu, and Ping Tan. Rcp: Recurrent closest point for point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8216–8226, 2022.

[9] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3254–3263, 2019.

[10] Zhangxuan Gu, Haoxing Chen, Zhuoer Xu, Jun Lan, Changhua Meng, and Weiqiang Wang. Diffusioninst: Diffusion model for instance segmentation. *arXiv preprint arXiv:2212.02773*, 2022.

[11] Xizewen Han, Huangjie Zheng, and Mingyuan Zhou. Card: Classification and regression diffusion models. *Advances in Neural Information Processing Systems*, 35:18100–18115, 2022.

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[13] Karl Holmquist and Bastian Wandt. Diffpose: Multi-hypothesis human pose estimation using diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15977–15987, 2023.

[14] Haobo Jiang, Mathieu Salzmann, Zheng Dang, Jin Xie, and Jian Yang. Se(3) diffusion model-based point cloud registration for robust 6d object pose estimation. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] Zhao Jin, Yinjie Lei, Naveed Akhtar, Haifeng Li, and Munawar Hayat. Deformation and correspondence aware unsupervised synthetic-to-real scene flow estimation for point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7233–7243, 2022.

[16] Philipp Jund, Chris Sweeney, Nichola Abdo, Zhifeng Chen, and Jonathon Shlens. Scalable scene flow from point clouds in the real world. *IEEE Robotics and Automation Letters*, 7(2):1589–1596, 2021.

[17] Yair Kittenplon, Yonina C Eldar, and Dan Raviv. Flowstep3d: Model unrolling for self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4114–4123, 2021.

[18] Bing Li, Cheng Zheng, Silvio Giancola, and Bernard Ghanem. Sctn: Sparse convolution-transformer network for scene flow estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1254–1262, 2022.

[19] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. *Advances in Neural Information Processing Systems*, 34:7838–7851, 2021.

[20] Xueqian Li, Jianqiao Zheng, Francesco Ferroni, Jhony Kaesemodel Pontes, and Simon Lucey. Fast neural scene flow. *arXiv preprint arXiv:2304.09121*, 2023.

[21] Jiuming Liu, Guangming Wang, Weicai Ye, Chaokang Jiang, Jinru Han, Zhe Liu, Guofeng Zhang, Dalong Du, and Hesheng Wang. Difflow3d: Toward robust uncertainty-aware scene flow estimation with diffusion

 https://doi.org/10.52202/079017-3532

model. *arXiv preprint arXiv:2311.17456*, 2023.

[22] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 529–537, 2019.

[23] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3614–3622, 2019.

[24] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016.

[25] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015.

[26] Gilles Puy, Alexandre Boulch, and Renaud Marlet. Flot: Scene flow on point clouds guided by optimal transport. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII*, pages 527–544. Springer, 2020.

[27] Zhile Ren, Deqing Sun, Jan Kautz, and Erik Sudderth. Cascaded scene flow prediction using semantic segmentation. In *2017 International Conference on 3D Vision (3DV)*, pages 225–233. IEEE, 2017.

[28] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[29] Saurabh Saxena, Charles Herrmann, Junhwa Hur, Abhishek Kar, Mohammad Norouzi, Deqing Sun, and David J Fleet. The surprising effectiveness of diffusion models for optical flow and monocular depth estimation. *Advances in Neural Information Processing Systems*, 36, 2024.

[30] Daniel Seita, Yufei Wang, Sarthak J Shetty, Edward Yao Li, Zackory Erickson, and David Held. Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds. In *Conference on Robot Learning*, pages 1038–1049. PMLR, 2023.

[31] Leonhard Sommer, Philipp Schröppel, and Thomas Brox. Sf2se3: Clustering scene flow into se (3)-motions via proposal and selection. In *Pattern Recognition: 44th DAGM German Conference, DAGM GCPR 2022, Konstanz, Germany, September 27–30, 2022, Proceedings*, pages 215–229. Springer, 2022.

[32] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *International Conference on Learning Representations*, 2021.

[33] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[34] Zachary Teed and Jia Deng. Raft-3d: Scene flow using rigid-motion embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8375–8384, 2021.

[35] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a piecewise rigid scene model. *International Journal of Computer Vision*, 115:1–28, 2015.

[36] Guangming Wang, Yunzhe Hu, Zhe Liu, Yiyang Zhou, Masayoshi Tomizuka, Wei Zhan, and Hesheng Wang. What matters for 3d scene flow network. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*, pages 38–55. Springer, 2022.

[37] Haiyan Wang, Jiahao Pang, Muhammad A Lodhi, Yingli Tian, and Dong Tian. Festa: Flow estimation via spatial-temporal attention for scene point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14173–14182, 2021.

[38] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[39] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 91–98, 2020.

[40] Yi Wei, Ziyi Wang, Yongming Rao, Jiwen Lu, and Jie Zhou. Pv-raft: Point-voxel correlation fields for scene flow estimation of point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6954–6963, 2021.

[41] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *European Conference on Computer Vision*, pages 88–107. Springer, 2020.

[42] Gengshan Yang and Deva Ramanan. Learning to segment rigid motions from two frames. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1266–1275, 2021.

[43] Yushan Zhang, Johan Edstedt, Bastian Wandt, Per-Erik Forssén, Maria Magnusson, and Michael Felsberg. Gmsf: Global matching scene flow. *Advances in Neural Information Processing Systems*, 36, 2024.

A Detailed model architecture

Feature Extraction In this section we give the detailed architecture of the feature extraction backbone, DGCNN [38], in Figure 5.

The overall architecture of DGCNN is given in the upper figure. The input point cloud ($\in \mathbb{R}^{N\times 3}$) contains the three-dimensional coordinates of N points. The whole network consists of four layers. The output feature of each layer, Feature 1 ($\in \mathbb{R}^{N\times 64}$), Feature 2 ($\in \mathbb{R}^{N\times 64}$), Feature 3 ($\in \mathbb{R}^{N\times 64}$), and Feature 4 ($\in \mathbb{R}^{N\times 128}$), are concatenated together along the feature dimension ($\in \mathbb{R}^{N\times 320}$) and then sent into a multi-layer perceptron (MLP) (linear layer + batch normalization + leaky ReLU) to get the final output feature ($\in \mathbb{R}^{N\times 128}$).

The detailed architecture of each layer is given in the bottom figure. $\mathbf{X}_{\text{input}} \in \mathbb{R}^{N\times C_{in}}$ is the input feature, where C_{in} is the number of input channels. A k-nearest-neighbor algorithm is first employed on the input feature to find the neighbors $\mathbf{X}_{\text{neighbor}} \in \mathbb{R}^{N\times k\times C_{in}}$. k is set to 16. The point feature $\mathbf{X}_{\text{input}}$ and the edge feature $\mathbf{X}_{\text{neighbor}} - \mathbf{X}_{\text{input}}$ are first concatenated together along the channel dimension ($\in \mathbb{R}^{N\times k\times 2C_{in}}$) and then sent into an MLP (linear layer + batch normalization + leaky ReLU). The output of the MLP has a dimension of $\mathbb{R}^{N\times k\times C_{out}}$, where C_{out} is the number of output channels. Finally, max pooling is done on the k-nearest-neighbors to get the final output feature ($\in \mathbb{R}^{N\times C_{out}}$).

Local Transformer The overall architecture of the local transformer is given in Figure 6 (left). The input features have a dimension of $\mathbb{R}^{N\times C_{in}}$. First, the input features are passed through a linear layer to get $\mathbf{X}_{\text{input}} \in \mathbb{R}^{N\times C_{in}}$. Then, the k-nearest-neighbors are found, $\mathbf{X}_{\text{neighbor}} \in \mathbb{R}^{N\times k\times C_{in}}$. The query is generated from $\mathbf{X}_{\text{input}}$ by linear layer φ, $\mathbf{Q} \in \mathbb{R}^{N\times C_q}$. The key and value are generated from $\mathbf{X}_{\text{neighbor}}$ by linear layers ψ and α, $\mathbf{K} \in \mathbb{R}^{N\times k\times C_k}$ and $\mathbf{V} \in \mathbb{R}^{N\times k\times C_v}$. $C_{in} = C_q = C_k = C_v = C_{out} = 128$. $k = 16$. γ is an MLP consisting of a sequence of linear layer, ReLU, linear layer, and softmax. The attention output ($\in \mathbb{R}^{N\times C_{out}}$) is further processed by a linear layer and a residual connection from the input.

The detailed positional encoding network is given in Figure 6 (right) which takes the coordinate of the input features as input $\mathbf{X}_{\text{input}} \in \mathbb{R}^{N\times 3}$. First, the k-nearest-neighbors of each point are found, $\mathbf{X}_{\text{neighbor}} \in \mathbb{R}^{N\times k\times 3}$. The distance between $\mathbf{X}_{\text{input}}$ and $\mathbf{X}_{\text{neighbor}}$ is passed through an MLP (linear layer + ReLU + linear layer) to get the final positional embedding δ.

Global Transformer In this section, we give the detailed architecture of the global transformer, in Figure 7 (left). The input features have a dimension of $\mathbf{X}_{\text{input}} \in \mathbb{R}^{N\times C_{in}}$. The query $\mathbf{Q} \in \mathbb{R}^{N\times C_q}$, key $\mathbf{K} \in \mathbb{R}^{N\times C_k}$, and value $\mathbf{V} \in \mathbb{R}^{N\times C_v}$ are generated from the input features by linear layers φ, ψ, and α. The attention matrix ($\in \mathbb{R}^{N\times N}$) is computed as the matrix multiplication of the query and the key. The attention output ($\in \mathbb{R}^{N\times C_v}$) is the matrix multiplication of the attention matrix and the value. The attention output is further processed by a linear layer, a layer normalization, and a residual connection from the input features to get the final output feature ($\in \mathbb{R}^{N\times C_{out}}$). $C_{in} = C_q = C_k = C_v = C_{out} = 128$.

Cross Transformer In this section, we give the detailed architecture of the cross transformer, in Figure 7 (right). $\mathbf{X}_{\text{source}} \in \mathbb{R}^{N\times C_{in}}$ and $\mathbf{X}_{\text{target}} \in \mathbb{R}^{N\times C_{in}}$ are the input features of the source point cloud and the input features of the target point cloud. The query $\mathbf{Q} \in \mathbb{R}^{N\times C_q}$ is generated from the source point cloud feature by linear layer φ. The key $\mathbf{K} \in \mathbb{R}^{N\times C_k}$ and value $\mathbf{V} \in \mathbb{R}^{N\times C_v}$ are generated from the target point cloud by linear layer ψ and α, respectively. Similarly to the global transformer, the attention matrix ($\in \mathbb{R}^{N\times N}$) is computed as the matrix multiplication of the query and the key. The attention output ($\in \mathbb{R}^{N\times C_v}$) is the matrix multiplication of the attention matrix and the value. The attention output is further processed by a linear layer, a layer normalization, and a residual connection from the input features to get the output feature ($\in \mathbb{R}^{N\times C_{out}}$). $C_{in} = C_q = C_k = C_v = C_{out} = 128$. A feedforward network with a sequence of linear layer, GeLU, linear layer, and layer normalization is applied to the output feature of the cross transformer to aggregate information. The final output has a dimension of $\mathbb{R}^{N\times C_{out}}$ The proposed method is trained on 4 $\times$ NVIDIA A40 GPUs.

https://doi.org/10.52202/079017-3532

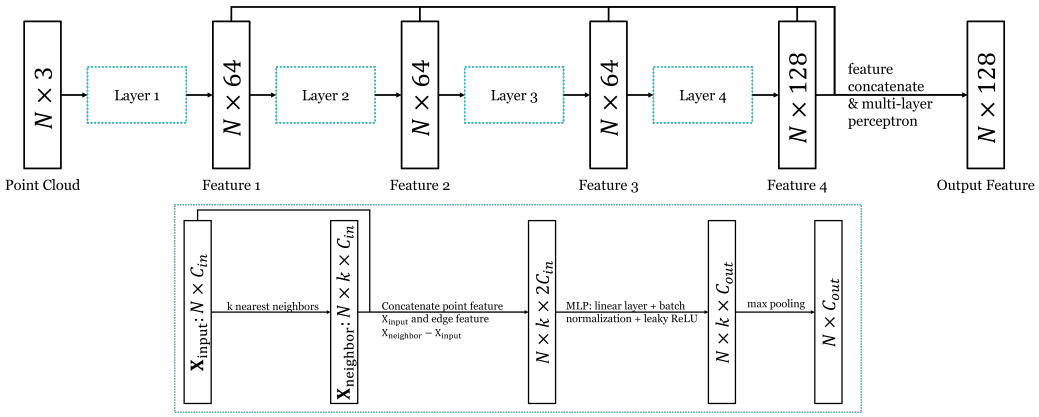

Figure 5: The detailed architecture of the feature extraction backbone DGCNN. The upper figure shows the overall architecture of DGCNN. The bottom figure shows the detailed architecture of each layer.

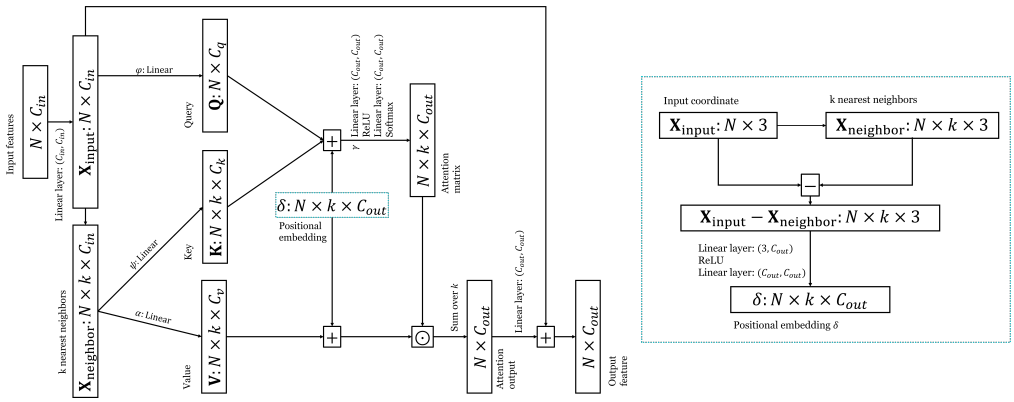

Figure 6: The detailed architecture of the local transformer. The left figure shows the overall architecture of the local transformer. The right figure shows the detailed architecture for the positional embedding.

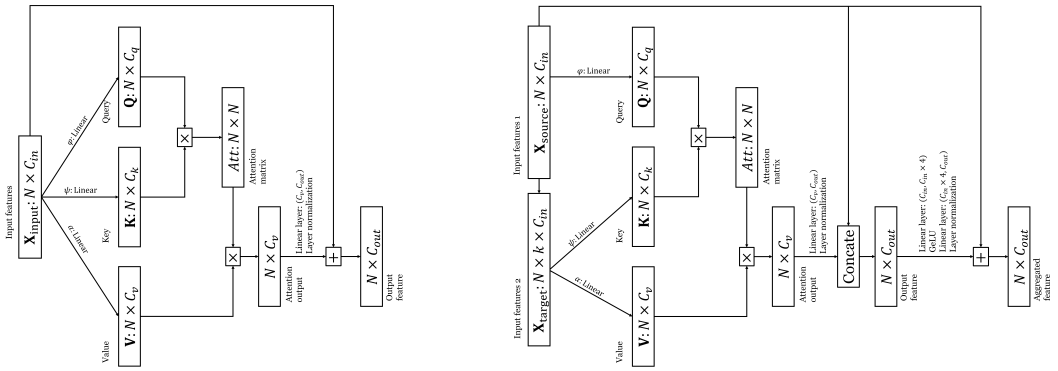

Figure 7: The detailed architecture of the global transformer (left) and the cross transformer (right).

B Additional visualizations

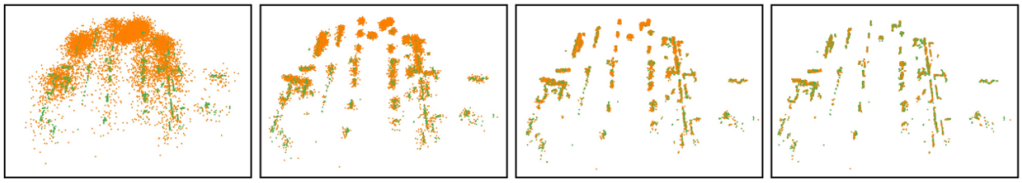

Figure 8: Visualization of the reverse diffusion process on the KITTI dataset. The orange points denote the source point cloud wrapped by the prediction of the current timestep. The green points denote the target point cloud.

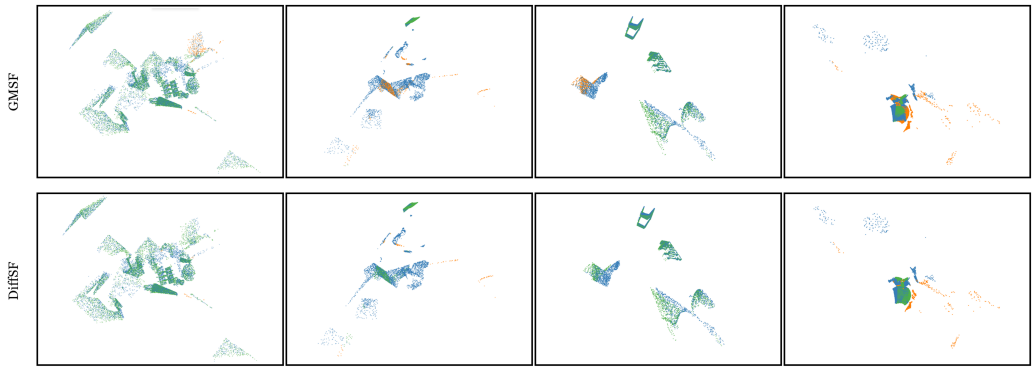

Figure 9: Visualization comparison of GMSF and DiffSF on the FlyingThings3D dataset. The blue points represent the target point cloud. The green points represent the warped source points with an EPE3D smaller than a certain threshold. The orange points represent the warped source points with an EPE3D larger than a certain threshold.

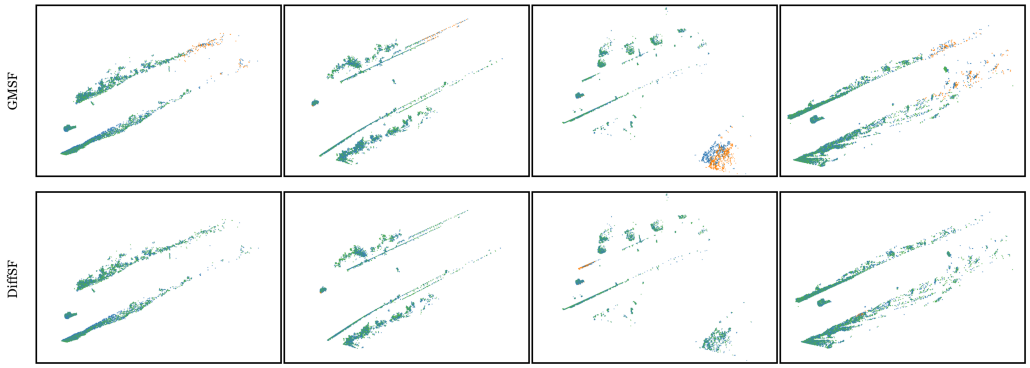

Figure 10: Visualization comparison of GMSF and DiffSF on the KITTI dataset. The blue points represent the target point cloud. The green points represent the warped source points with an EPE3D smaller than a certain threshold. The orange points represent the warped source points with an EPE3D larger than a certain threshold.

https://doi.org/10.52202/079017-3532

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations of the work can be found in the Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: There is no theoretical result in the paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The detailed experiment setting can be found in the experiments section 4. The detailed architecture of the model can be found in the Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

https://doi.org/10.52202/079017-3532

Answer: [Yes]

Justification: The code is available at https://github.com/ZhangYushan3/DiffSF, with sufficient instructions to faithfully reproduce the experimental results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The details can be found in the Experiment section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars are not reported because it would be too computationally expensive.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computer resources we used are specified in Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Societal impacts of the work can be found in the Section 5.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

https://doi.org/10.52202/079017-3532

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The work does not have a high risk for misuse such that safeguards are needed.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators or original owners of assets (e.g., code, data, models), used in the paper are properly credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets is introduced in the paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-3532