Self-Supervised Robust Scene Flow Estimation via the Alignment of Probability Density Functions

Pan He, Patrick Emami, Sanjay Ranka, Anand Rangarajan

Department of Computer and Information Science and Engineering, University of Florida
432 Newell Dr, Gainesville, FL 32611, USA
pan.he,pemami@ufl.edu; ranka,anand@cise.ufl.edu

Abstract

In this paper, we present a new self-supervised scene flow estimation approach for a pair of consecutive point clouds. The key idea of our approach is to represent discrete point clouds as continuous probability density functions using Gaussian mixture models. Scene flow estimation is therefore converted into the problem of recovering motion from the alignment of probability density functions, which we achieve using a closed-form expression of the classic Cauchy-Schwarz divergence. Unlike existing nearest-neighbor-based approaches that use hard pairwise correspondences, our proposed approach establishes soft and implicit point correspondences between point clouds and generates more robust and accurate scene flow in the presence of missing correspondences and outliers. Comprehensive experiments show that our method makes noticeable gains over the Chamfer Distance and the Earth Mover's Distance in real-world environments and achieves state-of-the-art performance among self-supervised learning methods on FlyingThings3D and KITTI, even outperforming some supervised methods with ground truth annotations.

Introduction

3D scene understanding (Qi et al. 2017a; Ilg et al. 2017; Liu, Qi, and Guibas 2019; Wang et al. 2019; Choy, Gwak, and Savarese 2019; He et al. 2021) of a dynamic environment has drawn increasing attention recently due to its wide applications in virtual reality, robotics, and autonomous driving. One fundamental task is *scene flow estimation* that aims at obtaining a 3D motion field of a dynamic scene (Vedula et al. 1999). Traditional scene flow methods focus on learning representations from stereo or RGB-D images (Basha, Moses, and Kiryati 2013; Jaimez et al. 2015; Teed and Deng 2021). Recently, researchers have started to design deep scene flow estimation networks for 3D point clouds (Gu et al. 2019; Liu, Qi, and Guibas 2019; Wu et al. 2020; Puy, Boulch, and Marlet 2020; Mittal, Okorn, and Held 2020; Gojcic et al. 2021; He et al. 2021).

However, major scene flow approaches rely on supervised learning with massive labeled training data that are expensive and difficult to obtain in real-world environments (Gu et al. 2019; Liu, Qi, and Guibas 2019; Puy, Boulch, and Marlet 2020). Consequently, researchers have turned to model training with synthetic data and rich annotations followed by a further fine-tuning step if necessary, or the use of self-supervised learning objectives to eliminate any dependence on labels. Early attempts at self-supervised scene flow estimation assume that the scene flow can be approximated by a point-wise transformation that moves the source point cloud to the target one (Wu et al. 2020; Mittal, Okorn, and Held 2020; Kittenplon, Eldar, and Raviv 2021). The alignment of point clouds is measured by popular similarity metrics such as the Chamfer Distance (CD) or Earth Mover's Distance (EMD). However, for scene flow estimation, these metrics are limited. CD is sensitive to outliers due to its nearest neighbor criterion and tends to obtain a degenerate solution as discussed in Mittal, Okorn, and Held (2020) and EMD is computationally heavy and its approximations can achieve poor performance in practice.

This paper presents a principled scene flow estimation objective that addresses both limitations; it is robust to missing correspondences and outliers *and* is efficient to compute. We accomplish this by proposing to represent discrete point clouds as continuous probability density functions (PDFs) using Gaussian mixture models (GMMs) and recovering motion by minimizing the divergence between two GMMs. This is in contrast to previous nearest-neighbor-based objectives which assume the existence of hard correspondences between pairs of discrete points. Intuitively, if point clouds are aligned well to each other, their resulting mixtures should be statistically similar. We, therefore, can obtain the approximated scene flow with a decent alignment between the source and target point clouds. In summary, our contributions are:

- A new perspective on self-supervised scene flow estimation as minimizing the divergence between two GMMs. The obtained soft correspondence between point cloud pairs differs from the existing nearest-neighbor-based approaches with the assumption of an explicit hard correspondence.
- A self-supervised objective that leverages the Cauchy-Schwarz divergence for aligning two GMMs. It admits an efficient closed-form expression and leads to more robust and accurate flow estimation over CD and EMD in the presence of missing correspondences and outliers on real-world datasets.

- State-of-the-art performance compared to other advanced self-supervised learning methods, even outperforming some fully-supervised models that use ground truth annotations.

Related Work

Distance Measures for Point Clouds

Here we provide an overview of some representative distance measures for point clouds.

Chamfer Distance: Given two point sets $\mathcal{P}_1$ and $\mathcal{P}_2$, the widely-used CD (Fan, Su, and Guibas 2017; Liu, Qi, and Guibas 2019) is one variant of the Hausdorff distance (Rockafellar and Wets 2009), which is defined as

$$\begin{aligned} D_{CD}(\mathcal{P}_1, \mathcal{P}_2) = & \frac{1}{|\mathcal{P}_1|} \sum_{x \in \mathcal{P}_1} \min_{y \in \mathcal{P}_2} d^2(x, y) \\ & + \frac{1}{|\mathcal{P}_2|} \sum_{y \in \mathcal{P}_2} \min_{x \in \mathcal{P}_1} d^2(x, y). \end{aligned} \tag{1}$$

Due to the unconstrained nearest neighbor search in CD, more than one point in $\mathcal{P}_1$ can link to the same point in $\mathcal{P}_2$ and vice versa. This many-to-one correspondence leads to noisy training signals due to the improper matching of outlier points, which are common in sparse and noisy LiDAR point clouds collected for popular autonomous driving datasets. Besides, as shown in Mittal, Okorn, and Held (2020), due to potentially large errors in scene flow prediction, given a source point p, estimated scene flow f_{est}, and ground truth scene flow f_{gt}, the nearest neighbor $\mathcal{N}(\tilde{p})$ of the translated point $\tilde{p} = p + f_{est}$ may not necessarily be the same as the real translated point $\hat{p} = p + f_{gt}$, implying that $\mathcal{N}(\tilde{p}) \not\equiv \hat{p}$ and noisy training signals indeed exist.

Earth Mover's Distance: A popular and efficient approximation of EMD (Rubner, Tomasi, and Guibas 2000; Fan, Su, and Guibas 2017) establishes a one-to-one correspondence, i.e., a bijection ϕ mapping $\mathcal{P}_1$ to $\mathcal{P}_2$, such that the sum of distances between *all* point pairs is minimal:

$$D_{EMD}(\mathcal{P}_1, \mathcal{P}_2) = \min_{\phi: \mathcal{P}_1 \to \mathcal{P}_2} \sum_{x \in \mathcal{P}_1} d(x, \phi(x)). \tag{2}$$

Obtaining an optimal mapping in EMD is non-trivial and suffers from high computational cost forcing researchers to introduce multiple approximations (Bertsekas 1992; Fan, Su, and Guibas 2017; Atasu and Mittelholzer 2019). Besides, because the approximate EMD focuses on a global point alignment (unlike D_{CD}), it tends to ignore local details when obtaining the optimal mapping—wrongly matching some points to distant points. Beyond these, other feature-based distance measures are explored for describing point cloud distances such as PointNet (Qi et al. 2017a) and 3DmFV (Ben-Shabat, Lindenbaum, and Fischer 2018).

As discussed above, both the CD and EMD can be sensitive to outliers and can amplify model estimation errors in a negative feedback loop during training. We address the limitations of these similarity measures by presenting a new differentiable objective function for scene flow estimation in the presence of noise and outliers.

End-to-End Scene Flow Estimation

Directly estimating scene flow from point cloud pairs using deep learning architectures has become a fast growing research area.

Fully-Supervised Approaches. FlowNet3D (Liu, Qi, and Guibas 2019) follows the PointNet++ architecture (Qi et al. 2017b) and introduces the *flow embedding* layer to aggregate spatio-temporal relationships of point sets based on feature similarities. HPLFlowNet (Gu et al. 2019) projects points onto permutohedral lattices and conducts bilateral convolutions for efficient processing, leading to competitive results. FLOT (Puy, Boulch, and Marlet 2020) follows optimal transport (Peyré, Cuturi et al. 2019) to establish soft point correspondences and further refine scene flow via a dynamic graph. FlowStep3D (Kittenplon, Eldar, and Raviv 2021) and PV-RAFT (Wei et al. 2021) iteratively refine scene flow predictions following (Teed and Deng 2020).

Self-Supervised Approaches. In (Mittal, Okorn, and Held 2020), they present a self-supervised scene flow approach that addresses some limitations of CD using cycle consistency, which is orthogonal to our objective. PointPWC-Net (Wu et al. 2020) takes inspiration from the *cost volume* for optical flow estimation (Sun et al. 2018) and generalizes the concept in a coarse-to-fine scene flow architecture. In (Pontes, Hays, and Lucey 2020), the authors regularize scene flows from point clouds via graph construction and application of the graph Laplacian. An adversarial learning approach for scene flow estimation (Zuanazzi et al. 2020) maps point clouds to a latent space in which a robust distance metric can be computed. Self-Point-Flow (Li, Lin, and Xie 2021) effectively leverages optimal transport (Peyré, Cuturi et al. 2019) to generate noisy pseudo ground truth and refines it via a random walk, in contrast to our end-to-end self-supervised objective with a closed-form expression.

Most existing self-supervised scene flow approaches (Wu et al. 2020; Mittal, Okorn, and Held 2020; Kittenplon, Eldar, and Raviv 2021) explicitly establish hard point correspondences between discrete point clouds based on the nearest neighbor criterion. Unlike them, we represent discrete point clouds as continuous PDFs, i.e., GMMs, and establish an implicit soft point correspondence via minimization of the divergence between the two corresponding mixtures. Our approach is less sensitive to the missing correspondences (e.g., due to occlusion or view changes) and outliers.

Point Set Registration

Kernel correlation-based registration was proposed in (Tsin and Kanade 2004) to align two point sets seen as PDFs. The method is further improved and generalized in (Roy, Gopinath, and Rangarajan 2007; Jian and Vemuri 2010; Hasanbelliu, Giraldo, and Príncipe 2011) with other divergences. Recently, deep learning methods have obtained impressive results for point set registration (Wang and Solomon 2019; Aoki et al. 2019; Yew and Lee 2020).

DeepGMR (Yuan et al. 2020) presents a deep registration method that minimizes the Kullback–Leibler divergence (Kullback and Leibler 1951) between point clouds. Our approach is related to these as we also represent discrete point clouds as PDFs to align them.

Problem Definition

Point clouds can represent raw data, e.g., 3D shapes, or the surfaces from which they are sampled, e.g., those collected or reconstructed from LiDAR or RGB-D sensors. Our goal is to estimate 3D scene flow from consecutive point cloud frames. Denote the source point cloud as $\boldsymbol{S} = \{(\boldsymbol{c_i^s}, \boldsymbol{x_i^s}) \mid i = 1, \ldots, N\}$ and target point cloud as $\boldsymbol{T} = \{(\boldsymbol{c_j^t}, \boldsymbol{x_j^t}) \mid j = 1, \ldots, M\}$, where $\boldsymbol{c_i^s}, \boldsymbol{c_j^t}$ are the 3D coordinates of individual points and $\boldsymbol{x_i^s}, \boldsymbol{x_j^t}$ are the associated point features, e.g., color or LiDAR intensity. Due to the viewpoint shift, occlusion and sampling effect, $\boldsymbol{S}$ and $\boldsymbol{T}$ do not necessarily have the same number of points or have strict point-to-point correspondences. Considering points $\boldsymbol{s_i} = (\boldsymbol{c_i^s}, \boldsymbol{x_i^s})$ in the source point cloud $\boldsymbol{S}$ being moved to a new location $\widehat{\boldsymbol{c_i^s}}$ at the target frame and denoting its 3D motion as $\boldsymbol{d_i} = \widehat{\boldsymbol{c_i^s}} - \boldsymbol{c_i^s}$, a scene flow estimation model will predict the motion for every sampled point $\boldsymbol{s_i}$ in the source point cloud $\boldsymbol{S}$ via a function f: $\boldsymbol{D} = \{\boldsymbol{d_i} = f(\boldsymbol{S}, \boldsymbol{T})_i \mid i = 1, \ldots, N\}$ such that they are close to real motion.

Proposed Approach

In this section, we introduce the proposed approach for representing and aligning discrete point clouds using PDFs. To the best of our knowledge, this is the first attempt at doing so for scene flow estimation.

Mixture Models for Representing Point Clouds

Unlike existing self-supervised learning objectives such as CD and EMD that rely on hard pairwise correspondences between discrete point clouds, the key idea of our paper is to represent point clouds by PDFs to obtain a soft correspondence. We demonstrate the conceptual differences between CD, EMD, and CS in Figure 2. The main intuition is that point clouds can be interpreted as samples drawn from continuous spatial distributions of point locations. By doing so, we can capture the uncertainty in point cloud generation, e.g., jitter introduced during the LiDAR scanning process.

Formally, for a given point cloud $\boldsymbol{x}$, we represent it as the PDF of a general Gaussian mixture, which is defined as $\mathcal{G}(x) = \sum_{k=1}^{K} w_k \mathcal{N}(x|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})$ with

$$\mathcal{N}(x|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}) = \frac{\exp\left[-\frac{1}{2}(x - \boldsymbol{\mu_k})^T \boldsymbol{\Sigma_k}^{-1}(x - \boldsymbol{\mu_k})\right]}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma_k}|}}, \quad (3)$$

where K is the number of Gaussian components. We denote w_k, μ_k, Σ_k as the mixture coefficient, mean, and covariance matrix of the k^{th} component of $\mathcal{G}(x)$. d is the feature dimension of each point. In our case, $d = 3$. $|\boldsymbol{\Sigma_k}| \equiv \det \boldsymbol{\Sigma_k}$ is the determinant of $\boldsymbol{\Sigma_k}$, also known as the generalized variance. Note that if K is large enough, $\mathcal{G}(x)$ can well approximate almost any underlying density of a point cloud.

Inspired by (Jian and Vemuri 2010; Roy, Gopinath, and Rangarajan 2007), we simplify the GMMs as follows: 1) the number of Gaussian components is the number of points with uniform weights (the occupancy probabilities or the mixture coefficients), 2) the mean vector of a component is the location of each point, and 3) all components share the same variance (isotropic, or spherical covariances), i.e., $\Sigma_i = \Sigma_j = \sigma \boldsymbol{I}$ with the identity matrix $\boldsymbol{I}$. We, therefore, obtain an overparameterized GMM model which can be equivalently obtained from a fixed-bandwidth kernel density estimation (KDE) with a Gaussian kernel (Scott 2015). More complicated GMMs are non-trivial and would require computationally expensive model fitting such as the Expectation-Maximization (EM) algorithm (Moon 1996), which we do not explore in this paper and instead reserve for future work.

Recovering Motion from the Alignment of PDFs

The principle of PDF divergence minimization results in the specification of a self-supervised learning objective that optimizes a scene flow model such that a dissimilarity measure $D_{dsim}(\mathcal{G}(\boldsymbol{S}_w), \mathcal{G}(\boldsymbol{T}))$ between the GMM representations of the warped point cloud $\mathcal{G}(\boldsymbol{S}_w) = \mathcal{G}(\boldsymbol{S} + \boldsymbol{D})$ and the target point cloud $\mathcal{G}(\boldsymbol{T})$ is minimized. Recall that $\boldsymbol{D} = \{\boldsymbol{d_i} = f(\boldsymbol{S}, \boldsymbol{T})_i \mid i = 1, \ldots, N\}$ where f is implemented as a deep neural network. We can construct a suitable D_{dsim} such that it is differentiable so it can guide optimization via backpropagation and gradient descent. We now describe how to achieve this goal.

We choose the Cauchy-Schwarz (CS) divergence (Jenssen et al. 2005; Principe 2010) for measuring the similarity between the two GMM representations of point clouds $\boldsymbol{S}_w$ and $\boldsymbol{T}$. The CS divergence can be *expressed in closed-form*, allowing an efficient end-to-end trainable implementation for scene flow estimation. We optimize f by minimizing

$$\begin{aligned}\mathcal{D}_{CS}(\mathcal{G}(\boldsymbol{S}_w), \mathcal{G}(\boldsymbol{T})) &= -\log\left(\frac{\int \mathcal{G}(\boldsymbol{S}_w)\mathcal{G}(\boldsymbol{T})dx}{\sqrt{\int \mathcal{G}^2(\boldsymbol{S}_w)dx \int \mathcal{G}^2(\boldsymbol{T})dx}}\right) \\ &= -\log \int \mathcal{G}(\boldsymbol{S}_w)\mathcal{G}(\boldsymbol{T})dx + 0.5 \log \int \mathcal{G}^2(\boldsymbol{S}_w)dx \\ &\quad + 0.5 \log \int \mathcal{G}^2(\boldsymbol{T})dx.\end{aligned} \quad (4)$$

The CS divergence is derived from the CS inequality (Steele 2004) and is expressed as inner products of PDFs. It defines a symmetric measure for any two PDFs $\mathcal{G}(\boldsymbol{S}_w)$ and $\mathcal{G}(\boldsymbol{T})$ such that $0 \leq D_{CS} < \infty$ where the minimum is obtained iff $\mathcal{G}(\boldsymbol{S}_w) = \mathcal{G}(\boldsymbol{T})$. It measures the interaction of the generated field of one PDF on the locations of the other PDF, which is also called the *cross information potential* of the two densities (Hasanbelliu, Giraldo, and Príncipe 2011).

Closed-form expression for GMMs: The CS divergence in Equation 4 can be written in a closed-form expression for GMMs (Jenssen et al. 2006). The basic idea is to follow the Gaussian identity (Petersen, Pedersen et al. 2008) to obtain the product of two Gaussian PDFs as

$$\begin{aligned}\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i})\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_j}, \boldsymbol{\Gamma_j}) = \\ \mathcal{N}(\boldsymbol{\mu_i}|\boldsymbol{\mu_j}, \boldsymbol{\Sigma_i} + \boldsymbol{\Gamma_j})\mathcal{N}(x|\boldsymbol{\mu_{ij}}, \boldsymbol{\Sigma_{ij}})\end{aligned} \quad (5)$$

where

$$\boldsymbol{\mu_{ij}} = \boldsymbol{\Sigma_{ij}}(\boldsymbol{\Sigma_i}^{-1}\boldsymbol{\mu_i} + \boldsymbol{\Gamma_j}^{-1}\boldsymbol{\mu_j}) \quad (6)$$

and

$$\boldsymbol{\Sigma_{ij}} = (\boldsymbol{\Sigma_i}^{-1} + \boldsymbol{\Gamma_j}^{-1})^{-1}. \quad (7)$$

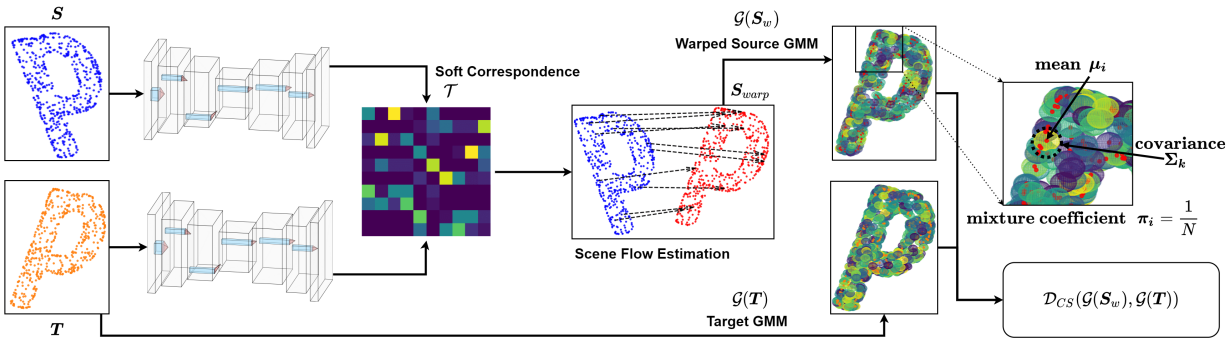

Figure 1: Overview of the proposed self-supervised learning for scene flow estimation. Our model takes both source and target point clouds to extract deep features via a UNet-like encoder-decoder backbone network (Ronneberger, Fischer, and Brox 2015) based on MinkowskiNet (Choy, Gwak, and Savarese 2019). We then warp the source point cloud by adding the estimated scene flow. Both the warped source and target point clouds are further fit using two separate GMMs. We train the model by minimizing the discrepancy between the two corresponding mixtures via a closed-form expression for the CS divergence.

Then we can use the Gaussian identity trick to simplify each term in the right of Equation 4 and get

$$\mathcal{D}_{CS}(\mathcal{G}(\boldsymbol{S}_w), \mathcal{G}(\boldsymbol{T})) = -\log\left(\sum_{i,j=1}^{N,M} \pi_i \tau_j \mathcal{N}(\boldsymbol{c}_i^s | \boldsymbol{c}_j^t, \boldsymbol{\Sigma_i} + \boldsymbol{\Gamma_j})\right)$$
$$+ 0.5 \log\left(\sum_{i,i'=1}^{N,N} \pi_i \pi_{i'} \mathcal{N}(\boldsymbol{c}_i^s | \boldsymbol{c}_{i'}^s, \boldsymbol{\Sigma_i} + \boldsymbol{\Sigma_{i'}})\right)$$
$$+ 0.5 \log\left(\sum_{j,j'=1}^{M,M} \tau_j \tau_{j'} \mathcal{N}(\boldsymbol{c}_j^t | \boldsymbol{c}_{j'}^t, \boldsymbol{\Gamma_j} + \boldsymbol{\Gamma_{j'}})\right), \tag{8}$$

where we denote the sets of mixture coefficients for two GMMs $\mathcal{G}(\boldsymbol{S}_w)$ and $\mathcal{G}(\boldsymbol{T})$ as $\{\pi_i\}_{i=1}^N$ and $\{\tau_j\}_{j=1}^M$ and the corresponding covariance matrix sets as $\{\Sigma_i\}_{i=1}^N$ and $\{\Gamma_j\}_{j=1}^M$. Note that the third term in the right of Equation 8 is a constant value for a target point cloud and can be optionally removed for faster computation. The detailed derivation of $\mathcal{D}_{CS}(\mathcal{G}(\boldsymbol{S}_w), \mathcal{G}(\boldsymbol{T}))$ can be found in the appendix.

Discussion: Examining $\mathcal{D}_{CS}(\mathcal{G}(\boldsymbol{S}_w), \mathcal{G}(\boldsymbol{T}))$, we can see that the exponential function in the Normal PDFs applied to the square of the point-pair distances weighed by a combination of Σ_i and Γ_j mitigates over-sensitivity to outliers by suppressing large squared distances. This is in contrast to the ℓ_2 distance-based approaches such as CD and EMD where outliers can contribute large values to the loss and negatively impact training. Also, it is fully differentiable and can be easily implemented with a few lines of code, which we provide in the appendix, in contrast to the sub-differentiable CD due to the min operator and the sub-optimal EMD approximation with expensive computation (Fan, Su, and Guibas 2017). Due to the probabilistic formulation, it supports the processing of point clouds with different sizes while being tolerant to noise. The conceptual difference between CD, EMD, and CS is illustrated in Figure 2. We note that CS is also closely related to graph cuts and Mercer kernel theory (see (Jenssen et al. 2006) for a detailed discussion).

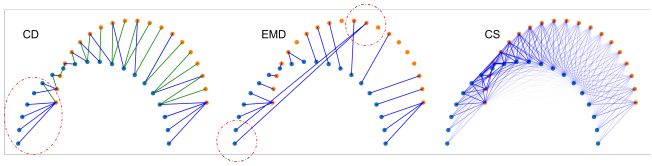

Figure 2: A toy example to illustrate the conceptual difference between CD, EMD, and CS. For CD and EMD, we visualize their matching correspondences between points of blue and orange curves by adding blue (the forward matching) and green (the backward matching) arrows. CD and the EMD approximation find the hard pairwise correspondence while (Fan, Su, and Guibas 2017) CS tries to link every blue point to all orange points via Gaussian functions — here we only show $\mathcal{N}(\boldsymbol{c}_i^s | \boldsymbol{c}_j^t, \boldsymbol{\Sigma_i} + \boldsymbol{\Gamma_j})$ in Equation 8 and weight the arrow color according to their values.

Model Implementation

We now demonstrate a deep neural network for self-supervised scene flow estimation that is trained with the CS divergence (Figure 1). It consists of : 1) deep feature extraction, 2) scene flow estimation from soft correspondences and 3) our CS divergence objective. We also add other regularization techniques to regularize the unconstrained scene flow predictions by encouraging rigid motion in local regions.

Deep Feature Extraction: To obtain features that encode raw point coordinates into higher dimension, we utilize a UNet-like encoder-decoder backbone network (Ronneberger, Fischer, and Brox 2015) based on MinkowskiNet (Choy, Gwak, and Savarese 2019). It receives inputs that are voxelized from point clouds. Denote these sparse voxels as $V^s \in \mathbb{R}^{N^s \times 3}$ and $V^t \in \mathbb{R}^{N^t \times 3}$ for $\mathcal{S}$ and $\mathcal{T}$ respectively. Both V^s and V^t will be feed into a same backbone with shared weights to obtain their corresponding deep features $F^s \in \mathbb{R}^{N^s \times 64}$ and $F^t \in \mathbb{R}^{N^t \times 64}$.

Scene Flow Estimation: Similar to (Puy, Boulch, and Marlet 2020; Wei et al. 2021), we construct a cost matrix $C \in$

$\mathbb{R}^{N^s \times N^t}$ and establish a soft-correspondence between sparse voxels. Formally, we compute a cost between every point $f_i^s \in F^s$ and every point $f_j^t \in F^t$ defined as

$$C_{ij} = 1 - \frac{\langle f_i^s, f_j^t \rangle}{\|f_i^s\|_2 \|f_j^t\|_2}. \tag{9}$$

Based on the cost matrix C, we find the top-K smallest values for each sparse voxel $v_i^s \in V^s$ regarding V^t, whose corresponding index set is denoted as $\mathcal{N}_i^k$. We then estimate the scene flow of v_i^s as

$$d_i^v = \frac{\sum_{j \in \mathcal{N}_i^k} \mathcal{T}_{ij} v_j^t}{\sum_{j \in \mathcal{N}_i^k} \mathcal{T}_{ij}} - v_i^s. \tag{10}$$

where $\mathcal{T}_{ij} = \exp(-C_{ij}/\epsilon)$. We empirically set $\epsilon = 0.00625$. Therefore, d_i^v is the weighted distance between v_i^s and its top-K points $\{v_j^t | j \in \mathcal{N}_i^k, |\mathcal{N}_i^k| \leq N^t\}$. We apply the inverse-distance weighted interpolation to transfer the flow for the sparse voxels to each point $s_i = \{c_i^s, x_i^s\}$ in S:

$$d_i^s = \frac{\sum_{j:v_j^s \in \mathcal{M}(s_i)} d_i^v \|v_j^s - c_i^s\|_2^{-1}}{\sum_{j:v_j^s \in \mathcal{M}(s_i)} \|v_j^s - c_i^s\|_2^{-1}}, \tag{11}$$

where $\mathcal{M}(s_i)$ finds the k-nearest neighbor (KNN) voxels of the point s_i based on the Euclidean distance.

Training

Our model is trained end-to-end without requiring any ground-truth scene flow annotations.

The CS divergence loss: We use the introduced CS divergence for self-supervised learning.

$$E_{cs}(\boldsymbol{D}) = \mathcal{D}_{CS}(\mathcal{G}(\boldsymbol{S}_w), \mathcal{G}(\boldsymbol{T})), \tag{12}$$

where we add the predicted scene flow $\boldsymbol{D}$ to the source point cloud $\mathcal{S}$ to obtain $\boldsymbol{S}_w$. It aligns the warped source and target point clouds.

The graph Laplacian loss: As each estimated scene flow vector in $\boldsymbol{D}$ has three degrees of freedom, only applying CS is under-constrained with many possible estimations. We further regularize them by adding an extra constraint to enforce the transformation to be as rigid as possible (Bobenko and Springborn 2007), which we formulate as

$$E_r(\boldsymbol{D}) = \sum_{\{i,j\} \in E} \|d_i - d_j\|_1, \tag{13}$$

where E defines the edge set of a graph G built upon the source point cloud $\mathcal{S}$. There exist various ways of constructing the graph: KNN graphs, fixed-Radius Nearest Neighbor graphs are possibilities. Following (Pontes, Hays, and Lucey 2020), we adopt the KNN graph due to its better sparsity and connectivity properties. We reformulate Equation 13 as

$$E_r(\boldsymbol{D}) = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{N} \frac{1}{|\mathcal{I}(s_i)|} \sum_{j \in \mathcal{I}(s_i)} \|d_i - d_j\|_1, \tag{14}$$

where $\mathcal{I}(s_i)$ denotes the index set of the k-nearest neighbor points of s_i in $\mathcal{S}$. We empirically set $k = 50$ leaving this choice up for future exploration.

The full objective function is a weighted sum of the CS divergence loss and the graph Laplacian loss:

$$E(\boldsymbol{D}) = E_{cs}(\boldsymbol{D}) + \lambda E_r(\boldsymbol{D}) \tag{15}$$

where λ is a hyperparameter to balance two terms. The first term minimizes the discrepancy between the mixtures representing the warped source point cloud $\mathcal{S} + \boldsymbol{D}$ and the target $\mathcal{T}$. The second term enforces the predicted flows of nearby points to be similar to each other.

Experiments

In this section, we describe the datasets and the associated evaluation metrics. We demonstrate the advantage of the proposed CS loss over CD and EMD. We achieve state-of-the-art performance compared to self-supervised approaches and even surpass some fully supervised methods.

Datasets and Evaluation Metrics

FlyingThings3D (FT3D): The dataset (Mayer et al. 2016) is the first large-scale synthetic dataset designed for scene flow estimation where each scene contains multiple randomly-moving objects taken from the ShapeNet dataset (Chang et al. 2015). Following (Liu, Qi, and Guibas 2019; Gu et al. 2019), we reconstructed point clouds and their ground truth scene flows, ending up with a total of 19,640 training examples and 3,824 test examples. We selected 3,928 examples from the training set for a hold-out validation.

KITTI Scene Flow (KSF): This real-world dataset (Menze, Heipke, and Geiger 2015; Menze and Geiger 2015) is adapted from the KITTI Scene Flow benchmark with 142 frame pairs. We obtained point clouds and ground truth scene flow by lifting the ground-truth disparity maps and optical flow to 3D (Gu et al. 2019). We removed ground points by heuristically setting a height threshold.

Unlabelled KITTI$_r$ Dataset: The KITTI$_r$ dataset is prepared by (Li, Lin, and Xie 2021) where they use raw point clouds in KITTI to produce a training dataset. It collects samples from those scenes not included in KSF and guarantees separate training and testing splits. It results in 6,068 training point cloud pairs sampled at every five frames.

Evaluation Metrics: We use the standard evaluation metrics (Liu, Qi, and Guibas 2019; Gu et al. 2019; Puy, Boulch, and Marlet 2020): 1) the *EPE3D[m]*, or the end-point error, which calculates the mean absolute distance error in meters, 2) the *strict ACC3D (Acc3DS)* considering the percentage of points whose $EPE3D[m] < 0.05m$ or relative error $< 5\%$, 3) the *relaxed ACC3D (Acc3DR)* considering the percentage of points whose $EPE3D[m] < 0.1m$ or relative error $< 10\%$, and 4) the *Outliers3D* to compute the percentage of points whose $EPE3D[m] > 0.3m$ or relative error $> 10\%$.

Implementation Details

We randomly sampled 8,192 points for each point cloud. All models were implemented in Pytorch (Paszke et al. 2019) and MinkowskiEngine (Choy, Gwak, and Savarese 2019). We utilized the Adam optimizer (Kingma and Ba 2014) with an initial learning rate of 0.001 and the cosine annealing scheduler. We trained models for 100 epochs and voxelized points at a resolution of 0.1m.

Method	Sup.	EPE3D [m]	Acc3DS	Acc3DR	Outliers
Flownet3D (2019)	*Full*	0.177	0.374	0.668	0.527
HPLFlowNet (2019)	*Full*	0.117	0.478	0.778	0.410
EgoFlow (2020)	*Full*	0.103	0.488	0.822	0.394
CD (Ours)	*Self*	0.170	0.477	0.697	0.470
EMD (Ours)	*Self*	0.192	0.426	0.666	0.503
CS (Ours)	*Self*	**0.105**	**0.633**	**0.832**	**0.338**

Table 1: Comparisons between models trained with different self-supervised objective functions. We train our models on KITTI$_r$ and evaluate them on KSF. Other methods are trained on FT3D with the fully-supervised learning. We find that CS is more robust to noise in LiDAR datasets than CD and EMD.

Comparison with CD and EMD

To verify the robustness of CS, we conduct one critical study of training models on KITTI$_r$. Specifically, to make a fair comparison, we use our model with the same architecture while applying different self-supervised objective functions including CD, EMD, and CS. We then evaluate these trained models on KSF with ground truth annotations. Because the KITTI$_r$ dataset is collected from autonomous vehicles across real-world environments with varied conditions, it tends to include more noise and outliers when compared to synthetic datasets. Therefore, we would expect models trained with a more robust objective function to achieve better performance.

As shown in Table 1, our model trained with CS outperforms both models with CD and EMD significantly. Specifically, it achieves a much lower EPE of 0.105, decreasing the errors of CD and EMD by 38.24% and 45.31%. It shows that CS is much more robust when conducting self-supervised learning on the real-world KITTI$_r$ dataset. Furthermore, it has achieved better performance compared to some representative state-of-the-art supervised methods, such as Flownet3D (Liu, Qi, and Guibas 2019), HPLFlowNet (Gu et al. 2019), and EgoFlow (Tishchenko et al. 2020).

Ablation Studies

Choices of the kernel bandwidth σ: Recall that the CS divergence with GMMs is closely related to fixed bandwidth KDE with Gaussian kernels. The KDE bandwidth equals the scale parameter (standard deviation) of a Gaussian kernel. It is important to choose a suitable bandwidth (corresponding to the square root of the isotropic variance in GMM models)—either too large or too small bandwidth values will not best approximate the true underlying density, resulting in degraded performance.

As shown in Figure 3, we plot the results by choosing different variances $\sigma^2 = 0.1, 0.01, 0.001, 0.0001$ where σ is the kernel bandwidth. When setting a large variance such as $\sigma^2 = 0.1$, it leads to *oversmoothing* where some important local structures, e.g., curvatures of a shape, are ignored due to a large amount of smoothing. A small variance such as $\sigma^2 = 0.0001$ tends to generate darting structures on a density curve which is sensitive to noise—considering its extreme situation ($\sigma \to 0$) that changes a Gaussian PDF to a Dirac delta function. The results show that choosing $\sigma^2 = 0.01$ or $\sigma^2 = 0.001$ leads to good results. Nevertheless, such a bandwidth selection can vary across datasets depending on the underlying data characteristics. We also explored conventional bandwidth selection methods including *Silverman's rule of thumb* (Silverman 2018) and *Improved Sheater-Jones (ISJ)* (Botev, Grotowski, and Kroese 2010). They achieve worse results, which might be caused by the improper data assumption, e.g., unimodal distribution in Silverman's rule, and uncertainty not being captured.

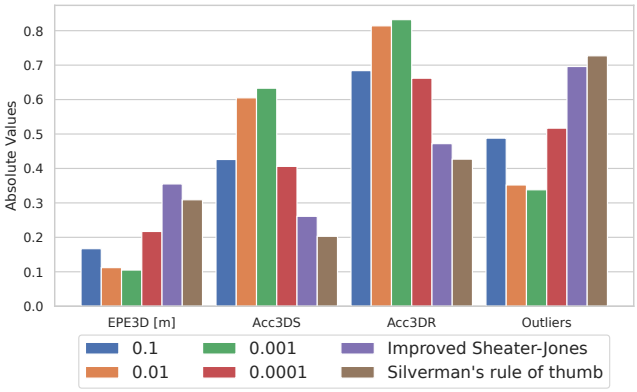

Figure 3: Results of CS with different kernel bandwidths. Choosing a suitable bandwidth leads to best performance.

	EPE3D [m] ↓	Acc3DS ↑	Acc3DR ↑	Outliers ↓
w/o (λ=0)	0.105	0.633	0.832	0.338
$\lambda = 100$	0.109	0.639	0.824	0.332
$\lambda = 10$	**0.096**	**0.686**	**0.855**	**0.302**
$\lambda = 1$	0.098	0.659	0.853	0.316
$\lambda = 0.1$	0.103	0.631	0.840	0.331

Table 2: Impact of graph Laplacian regularization. All model variants are trained on KITTI$_r$. Choosing a proper λ leads to further improvements.

Impact of graph Laplacian regularization: Encouraging rigid scene estimation via proper regularization is a critical step in handling datasets with many rigid objects. *Under-regularization* with an inadequately small λ has nearly no impact on original estimation with many possible predictions. *Over-regularization* with an excessively large λ adds over-strict constraints and leads to imperfect alignment between GMMs. Results are summarized in Table 2. Choosing a proper λ leads to further improvements on scene flow prediction, resulting in an roughly increase of 5% and 3% in absolute accuracy on Acc3DS and Acc3DR and about a decrease of 3% in absolute error on Outliers.

Evaluation on FlyingThings3D

We demonstrate its effectiveness by training on the FT3D dataset without using any ground truth annotations. Table 3 summarizes the evaluation results on the test split of the

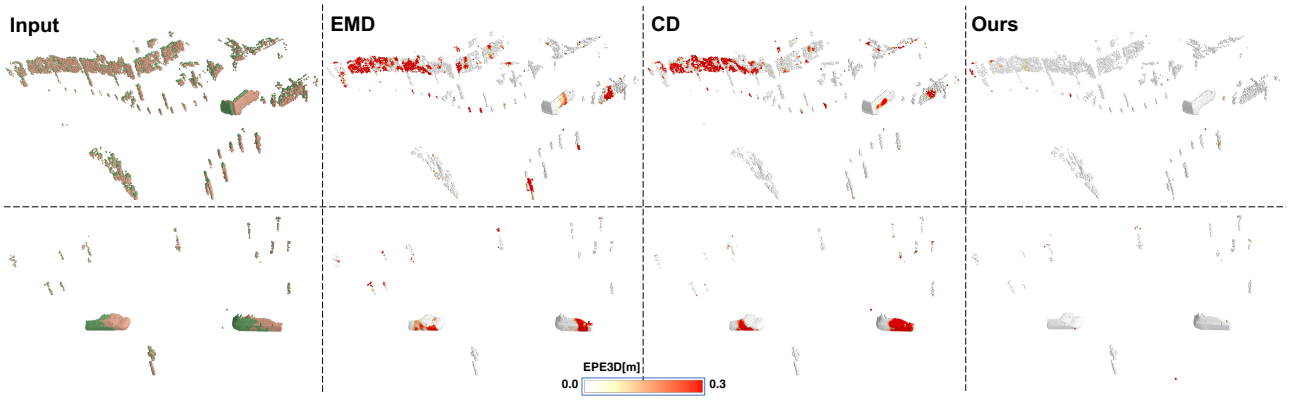

Figure 4: Qualitative results of our method on *KSF*. We clip the EPE3D[m] to the range between 0.0 m (white) and 0.3 (red). It confirms that our method is better on handling outliers compared to EMD and CD.

Method	Sup.	EPE3D[m]	Acc3DS	Acc3DR	Outliers
FlowNet3D (2019)	*Full*	0.114	0.412	0.771	0.602
HPLFlowNet (2019)	*Full*	0.080	0.614	0.855	0.429
PointPWC-Net (2020)	*Full*	0.059	0.738	0.928	0.342
FLOT (2020)	*Full*	0.052	0.732	0.927	0.357
EgoFlow (2020)	*Full*	0.069	0.670	0.879	0.404
R3DSF (2021)	*Full*	0.052	0.746	0.936	0.361
PV-RAFT (2021)	*Full*	0.046	0.817	0.957	0.292
FlowStep3D (2021)	*Full*	0.046	0.816	0.961	0.217
ICP (rigid) (1992)	*Self*	0.406	0.161	0.304	0.880
FGR (rigid) (2016)	*Self*	0.402	0.129	0.346	0.876
CPD (non-rigid) (2010)	*Self*	0.489	0.054	0.169	0.906
EgoFlow (2020)	*Self*	0.170	0.253	0.550	0.805
PointPWC-Net (2020)	*Self*	0.121	0.324	0.674	0.688
FlowStep3D (2021)	*Self*	0.085	0.536	0.826	**0.420**
Ours	*Self*	**0.075**	**0.589**	**0.862**	0.470

Table 3: Evaluation results on the *FT3D* datasets.

Method	Sup.	EPE3D[m]	Acc3DS	Acc3DR	Outliers
Flownet3D (2019)	*Full*	0.177	0.374	0.668	0.527
HPLFlowNet (2019)	*Full*	0.117	0.478	0.778	0.410
PointPWC-Net (2020)	*Full*	0.069	0.728	0.888	0.265
FLOT (2020)	*Full*	0.056	0.755	0.908	0.242
EgoFlow (2020)	*Full*	0.103	0.488	0.822	0.394
R3DSF (2021)	*Full*	0.042	0.849	0.959	0.208
PV-RAFT (2021)	*Full*	0.056	0.823	0.937	0.216
FlowStep3D (2021)	*Full*	0.055	0.805	0.925	0.149
ICP(rigid) (1992)	*Self*	0.518	0.067	0.167	0.871
FGR(rigid) (2016)	*Self*	0.484	0.133	0.285	0.776
CPD (non-rigid) (2010)	*Self*	0.414	0.206	0.400	0.715
EgoFlow (2020)	*Self*	0.415	0.221	0.372	0.810
PointPWC-Net (2020)	*Self*	*0.255*	*0.238*	*0.496*	*0.686*
FlowStep3D (2021)	*Self*	0.102	0.708	0.839	0.246
Ours	*Self*	**0.092**	**0.747**	**0.870**	0.283
Ours	*Self**	**0.096**	0.686	**0.855**	0.302

* denotes models trained with KITTI_r

Table 4: Evaluation results on the KITTI scene flow datasets.

FT3D dataset. Due to the better model design and objective function, our method outperforms all recent self-supervised frameworks including PointPWC-Net (Wu et al. 2020) and FlowStep3D (Kittenplon, Eldar, and Raviv 2021) and some full-supervised methods such as Flownet3D (Liu, Qi, and Guibas 2019) and HPLFlowNet (Gu et al. 2019). Specifically, among all self-supervised approaches, our RSFNet obtains the best values on $EPE3D[m]$, $Acc3DS$, and $Acc3DR$, decreasing the $EPE3D[m]$ of the previous best model (FlowStep3D) by 11.76% and increasing $Acc3DS$ and $Acc3DR$ of it by 9.88% and 4.36%. Note that FlowStep3D has conducted multiple inference steps to iteratively refine the scene flow prediction while our method is more efficient with one-step inference (see the appendix).

Evaluation on KITTI Scene Flow

We apply the trained model from the FT3D dataset to the unseen KSF dataset. As shown in Table 4, our method shows a good generalization result — achieving a lower 3D endpoint error 0.092 and higher accuracy (*near 4% absolute accuracy improvement over (Kittenplon, Eldar, and Raviv 2021)*). Models from the KITTI_r also perform competitively though trained on a much smaller dataset containing 6068 training samples, in contrast to 15,712 training samples in the FT3D, which shows a promising direction to close the synthetic-to-real gap with real-world training data. More results and details can be found in the appendix.

Conclusions

In this work, we have presented a novel self-supervised scene flow estimation approach that represents discrete point clouds as continuous Gaussian mixture PDFs and recovers motion from their alignment. Models trained with CS show more robustness and higher accuracy compared to CD and EMD. The resulting models have achieved state-of-the-art performance on standard datasets. We are encouraged by this to attempt the design of advanced iterative refinement techniques for better scene flow estimation in future work.

Acknowledgements

This work is supported by NSF CNS 1922782. The opinions, findings and conclusions expressed in this publication are those of the author(s) and not necessarily those of the National Science Foundation.

References

Aoki, Y.; Goforth, H.; Srivatsan, R. A.; and Lucey, S. 2019. PointNetLK: Robust & Efficient Point Cloud Registration using PointNet. In *IEEE Conference on Computer Vision and Pattern Recognition*, 7163–7172. IEEE.

Atasu, K.; and Mittelholzer, T. 2019. Linear-complexity data-parallel earth mover's distance approximations. In *International Conference on Machine Learning*, 364–373. PMLR.

Basha, T.; Moses, Y.; and Kiryati, N. 2013. Multi-view Scene Flow Estimation: A View Centered Variational Approach. *International Journal of Computer Vision*, 101(1): 6–21.

Ben-Shabat, Y.; Lindenbaum, M.; and Fischer, A. 2018. 3DmFV: Three-Dimensional Point Cloud Classification in Real-Time using Convolutional Neural Networks. *IEEE Robotics and Automation Letters*, 3(4): 3145–3152.

Bertsekas, D. P. 1992. Auction Algorithms for Network Flow Problems: A Tutorial Introduction. *Computational optimization and applications*, 1(1): 7–66.

Besl, P. J.; and McKay, N. D. 1992. Method for Registration of 3-D Shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, 586–606. International Society for Optics and Photonics.

Bobenko, A. I.; and Springborn, B. A. 2007. A Discrete Laplace–Beltrami Operator for Simplicial Surfaces. *Discrete & Computational Geometry*, 38(4): 740–756.

Botev, Z. I.; Grotowski, J. F.; and Kroese, D. P. 2010. Kernel Density Estimation via Diffusion. *The Annals of Statistics*, 38(5): 2916–2957.

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. 2015. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*.

Choy, C.; Gwak, J.; and Savarese, S. 2019. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 3075–3084.

Fan, H.; Su, H.; and Guibas, L. J. 2017. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *IEEE conference on Computer Vision and Pattern Recognition*, 605–613.

Gojcic, Z.; Litany, O.; Wieser, A.; Guibas, L. J.; and Birdal, T. 2021. Weakly Supervised Learning of Rigid 3D Scene Flow. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5692–5703.

Gu, X.; Wang, Y.; Wu, C.; Lee, Y. J.; and Wang, P. 2019. HPLFlowNet: Hierarchical Permutohedral Lattice FlowNet for Scene Flow Estimation on Large-scale Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, 3254–3263.

Hasanbelliu, E.; Giraldo, L. S.; and Príncipe, J. C. 2011. A Robust Point Matching Algorithm for Non-Rigid Registration using the Cauchy-Schwarz Divergence. In *IEEE International Workshop on Machine Learning for Signal Processing*, 1–6. IEEE.

He, P.; Emami, P.; Ranka, S.; and Rangarajan, A. 2021. Learning Scene Dynamics from Point Cloud Sequences. *arXiv preprint arXiv:2111.08755*.

Ilg, E.; Mayer, N.; Saikia, T.; Keuper, M.; Dosovitskiy, A.; and Brox, T. 2017. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. In *IEEE conference on Computer Vision and Pattern Recognition*, 2462–2470.

Jaimez, M.; Souiai, M.; Stückler, J.; Gonzalez-Jimenez, J.; and Cremers, D. 2015. Motion Cooperation: Smooth Piecewise Rigid Scene Flow from RGB-D Images. In *International Conference on 3D Vision*, 64–72. IEEE.

Jenssen, R.; Erdogmus, D.; Hild, K. E.; Principe, J. C.; and Eltoft, T. 2005. Optimizing the Cauchy-Schwarz PDF Distance for Information Theoretic, Non-Parametric Clustering. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 34–45.

Jenssen, R.; Principe, J. C.; Erdogmus, D.; and Eltoft, T. 2006. The Cauchy–Schwarz divergence and Parzen windowing: Connections to Graph Theory and Mercer Kernels. *Journal of the Franklin Institute*, 343(6): 614–629.

Jian, B.; and Vemuri, B. C. 2010. Robust Point Set Registration using Gaussian Mixture Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8): 1633–1645.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Kittenplon, Y.; Eldar, Y. C.; and Raviv, D. 2021. FlowStep3D: Model Unrolling for Self-Supervised Scene Flow Estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4114–4123.

Kullback, S.; and Leibler, R. A. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1): 79–86.

Li, R.; Lin, G.; and Xie, L. 2021. Self-Point-Flow: Self-Supervised Scene Flow Estimation from Point Clouds with Optimal Transport and Random Walk. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15577–15586.

Liu, X.; Qi, C. R.; and Guibas, L. J. 2019. FlowNet3D: Learning Scene Flow in 3D Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, 529–537.

Mayer, N.; Ilg, E.; Hausser, P.; Fischer, P.; Cremers, D.; Dosovitskiy, A.; and Brox, T. 2016. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *IEEE conference on Computer Vision and Pattern Recognition*, 4040–4048.

Menze, M.; and Geiger, A. 2015. Object Scene Flow for Autonomous Vehicles. In *IEEE Conference on Computer Vision and Pattern Recognition*, 3061–3070.

Menze, M.; Heipke, C.; and Geiger, A. 2015. Joint 3D Estimation of Vehicles and Scene Flow. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2: 427.

Mittal, H.; Okorn, B.; and Held, D. 2020. Just Go with the Flow: Self-Supervised Scene Flow Estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11177–11185.

Moon, T. K. 1996. The Expectation-Maximization Algorithm. *IEEE Signal Processing Magazine*, 13(6): 47–60.

Myronenko, A.; and Song, X. 2010. Point Set Registration: Coherent Point Drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12): 2262–2275.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, 8026–8037.

Petersen, K. B.; Pedersen, M. S.; et al. 2008. The Matrix Cookbook. *Technical University of Denmark*, 7(15): 510.

Peyré, G.; Cuturi, M.; et al. 2019. Computational Optimal Transport: With Applications to Data Science. *Foundations and Trends in Machine Learning*, 11(5-6): 355–607.

Pontes, J. K.; Hays, J.; and Lucey, S. 2020. Scene Flow from Point Clouds with or without Learning. *arXiv preprint arXiv:2011.00320*.

Principe, J. C. 2010. *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*. Springer Science & Business Media.

Puy, G.; Boulch, A.; and Marlet, R. 2020. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. In *European Conference on Computer Vision*, 527–544. Springer.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 652–660.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems*, 5099–5108.

Rockafellar, R. T.; and Wets, R. J.-B. 2009. *Variational analysis*, volume 317. Springer Science & Business Media.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 234–241. Springer.

Roy, A. S.; Gopinath, A.; and Rangarajan, A. 2007. Deformable Density Matching for 3D Non-Rigid Registration of Shapes. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 942–949.

Rubner, Y.; Tomasi, C.; and Guibas, L. J. 2000. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2): 99–121.

Scott, D. W. 2015. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons.

Silverman, B. W. 2018. *Density Estimation for Statistics and Data Analysis*. Routledge.

Steele, J. M. 2004. *The Cauchy-Schwarz Master Class: An Introduction to the Art of Mathematical Inequalities*. Cambridge University Press.

Sun, D.; Yang, X.; Liu, M.-Y.; and Kautz, J. 2018. PWC-Net: CNNs for Optical Flow using Pyramid, Warping, and Cost Volume. In *IEEE Conference on Computer Vision and Pattern Recognition*, 8934–8943.

Teed, Z.; and Deng, J. 2020. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *European Conference on Computer Vision*, 402–419. Springer.

Teed, Z.; and Deng, J. 2021. RAFT-3D: Scene Flow using Rigid-Motion Embeddings. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8375–8384.

Tishchenko, I.; Lombardi, S.; Oswald, M. R.; and Pollefeys, M. 2020. Self-Supervised Learning of Non-Rigid Residual Flow and Ego-Motion. In *International Conference on 3D Vision*, 150–159.

Tsin, Y.; and Kanade, T. 2004. A Correlation-Based Approach to Robust Point Set Registration. In *European conference on Computer Vision*, 558–569.

Vedula, S.; Baker, S.; Rander, P.; Collins, R.; and Kanade, T. 1999. Three-Dimensional Scene Flow. In *IEEE International Conference on Computer Vision*, 722–729.

Wang, Y.; and Solomon, J. M. 2019. Deep Closest Point: Learning Representations for Point Cloud Registration. In *IEEE/CVF International Conference on Computer Vision*, 3523–3532.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics*.

Wei, Y.; Wang, Z.; Rao, Y.; Lu, J.; and Zhou, J. 2021. PV-RAFT: Point-Voxel Correlation Fields for Scene Flow Estimation of Point Clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6954–6963.

Wu, W.; Wang, Z. Y.; Li, Z.; Liu, W.; and Fuxin, L. 2020. PointPWC-Net: Cost Volume on Point Clouds for (Self-) Supervised Scene Flow Estimation. In *European Conference on Computer Vision*, 88–107. Springer.

Yew, Z. J.; and Lee, G. H. 2020. RPM-Net: Robust Point Matching using Learned Features. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11824–11833.

Yuan, W.; Eckart, B.; Kim, K.; Jampani, V.; Fox, D.; and Kautz, J. 2020. DeepGMR: Learning Latent Gaussian Mixture Models for Registration. In *European Conference on Computer Vision*, 733–750. Springer.

Zhou, Q.-Y.; Park, J.; and Koltun, V. 2016. Fast Global Registration. In *European Conference on Computer Vision*, 766–782. Springer.

Zuanazzi, V.; van Vugt, J.; Booij, O.; and Mettes, P. 2020. Adversarial Self-Supervised Scene Flow Estimation. In *2020 International Conference on 3D Vision*, 1049–1058. IEEE.