Self-Supervised Class-Agnostic Motion Prediction with Spatial and Temporal Consistency Regularizations

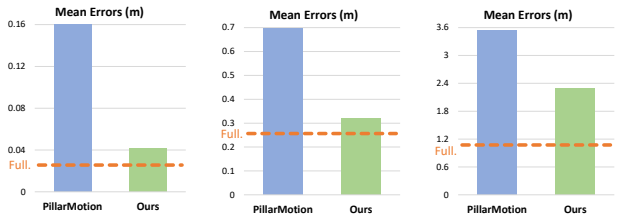
Kewei Wang[1,2], Yizheng Wu[1,2], Jun CEN[2], Zhiyu Pan[1], Xingyi Li[1,2], Zhe Wang[3], Zhiguo Cao[1], Guosheng Lin[2*]
[1] School of AIA, Huazhong University of Science and Technology
[2] S-Lab, Nanyang Technological University [3] SenseTime Research

Abstract

The perception of motion behavior in a dynamic environment holds significant importance for autonomous driving systems, wherein class-agnostic motion prediction methods directly predict the motion of the entire point cloud. While most existing methods rely on fully-supervised learning, the manual labeling of point cloud data is laborious and time-consuming. Therefore, several annotation-efficient methods have been proposed to address this challenge. Although effective, these methods rely on weak annotations or additional multi-modal data like images, and the potential benefits inherent in the point cloud sequence are still underexplored. To this end, we explore the feasibility of self-supervised motion prediction with only unlabeled LiDAR point clouds. Initially, we employ an optimal transport solver to establish coarse correspondences between current and future point clouds as the coarse pseudo motion labels. Training models directly using such coarse labels leads to noticeable spatial and temporal prediction inconsistencies. To mitigate these issues, we introduce three simple spatial and temporal regularization losses, which facilitate the self-supervised training process effectively. Experimental results demonstrate the significant superiority of our approach over the state-of-the-art self-supervised methods. Code will be available at https://github.com/kwwcv/SelfMotion.

1. Introduction

Motion perception provides key information for autonomous driving. Classical approaches formulate the motion perception task as object-level trajectory prediction [3, 5, 7, 17, 34]. However, these approaches may fail if agents are not successfully detected by the detectors, especially when encountering unseen categories in open-set scenarios [31]. To compensate for trajectory prediction and provide backup, class-agnostic motion prediction was studied [15, 25, 29–31]. Currently, these approaches predict the motions of the entire environment from point clouds directly instead of for each agent.



(a) Static (b) Slow (c) Fast

Figure 1. Performance comparison over static, slow, and fast speed levels between self-supervised PillarMotion [19] and our approach on the nuScenes dataset. The dashed line represents the performance of fully-supervised MotionNet [31]. Our proposed self-supervised approach outperforms the PillarMotion which uses additional image data by a large margin and substantially narrows the performance gap with fully-supervised results.

While fully-supervised class-agnostic motion prediction methods have shown success, they typically require a vast amount of annotated point cloud data, which can be both costly and time-consuming to acquire. To overcome this limitation, WeakMotionNet [15] proposes a weakly supervised approach using only a limited number of foreground-background semantic annotations. WeakMotionNet also introduces a consistency-aware chamfer loss that uses the difference between correspondences from different frames as loss weights. Although effective, it requires establishing correspondences multiple times and is still based on matching results between individual frames, rather than taking advantage of the temporal information within the input sequences. Without using any annotations, PillarMotion [19] introduces a self-supervised training approach that incorporates regularization of 2D and 3D flows derived from paired image and point cloud data. However, PillarMotion relies on multi-modality pairs to regularize motion learning, making the training process less flexible, and a considerable performance gap still exists between self-supervised

*Corresponding author.
gslin@ntu.edu.sg, {wangkewei, zgcao}@hust.edu.cn,

and fully-supervised methods. Therefore, we explore the spatial and temporal information from the point cloud sequences themselves to train a better class-agnostic motion prediction model with only unlabeled LiDAR data in this paper.

Our approach primarily involves pseudo label generation and spatial and temporal regularizations. To generate pseudo motion labels, we first solve a matching problem to find correspondence between current and future point clouds. However, the correctness of the matching correspondence cannot be guaranteed due to the occlusion, background artifacts, noise, etc. Self-supervised learning from these coarse matching correspondences (pseudo labels) leads to two major failure cases: (1) predicted motions from the same rigid object are spatially diverse; (2) the background cells that should be static are predicted with large motions. To alleviate these issues, we introduce three simple but effective spatial and temporal regularizations from the point cloud motion prediction itself to facilitate self-supervised learning. Specifically, cluster consistency regularization, backward consistency regularization, and forward consistency regularization are proposed to encourage spatial consistency, penalize incorrect predictions (*e.g.*, large motion for static cells), and smooth the prediction learned from pseudo labels of different time horizons, respectively.

Experiments demonstrate that our proposed spatial and temporal consistency regularization terms work effectively together, significantly boosting the performance of self-supervised motion prediction. The proposed approach turns out with excellent performance, as illustrated in Fig. 1. On the nuScenes dataset, our approach outperforms previous methods by a substantial margin. Our contributions can be summarized as follows:

- We propose a novel approach for self-supervised class-agnostic motion prediction that incorporates an optimal transport solver and three simple yet effective spatial and temporal regularization losses.
- Our approach is model-independent, and does not require any additional models or multi-modal data.
- Our method surpasses the state-of-the-art (SOTA) self-supervised methods by a large margin and significantly reduces the performance gap with the fully-supervised methods.

2. Related Works

Class-agnostic motion prediction. Classical motion prediction approaches aim to predict the future trajectories of agents based on past observations. The entire system typically involves object detection [11, 20, 35], tracking, and forecasting [3, 5, 7, 17, 33, 34]. However, the trajectory prediction relies on the object detector which may fail to detect objects with unknown classes in open-set scenarios [31]. To enhance system redundancy, especially in cases where objects may be not successfully detected and potentially lead to a failure in object-level trajectory forecasting [31], class-agnostic motion prediction methods predict the motion of entire scenes directly instead of individually for each agent. These methods represent the environment state based on BEV maps discretized from point clouds and aim to predict the 2D displacement vector for each BEV cell along the ground plane. MotionNet [31] is one of the pioneers in this task, proposing to perform joint perception and motion prediction based on a spatial-temporal pyramid network. LSTM-ED [25] introduce convolutional LSTM [26] to aggregate temporal context. BE-STI [30] leverages semantic cues by training an additional semantic decoder to guide motion prediction.

To alleviate the reliance on manual annotations, WeakMotionNet [15] proposes to train the MotionNet in a weakly-supervised manner with limited foreground and background semantic annotations. PillarMotion [19] proposes to train the model in a self-supervised manner without any annotations by cross-sensor motion regularization. Although effective, a substantial performance gap persists between the self-supervised PillarMotion and the fully-supervised model even using additional image data. Therefore, an open question remains about whether we can more efficiently train a self-supervised motion prediction model with higher performance.

Scene flow estimation. Scene flow estimation [1, 8, 18, 28, 32] from point clouds is an alternative way to understand the dense class-agnostic 3D motion field. In self-supervised scene flow estimation, several methods rely on establishing point correspondences between source and target point clouds through point matching. They treat the 3D coordinate difference between each matching pair as a pseudo scene flow label [10, 13, 14, 21, 24, 32]. Many methods also utilize the chamfer distance loss [9] to drive self-supervised scene flow learning [6, 10, 16, 23, 32].

While similar to the class-agnostic motion prediction task, scene flow estimation majorly differs in the following ways: (1) The objectiveness of the two tasks is different [15, 31]. Scene flow estimation methods focus on estimating the motion flow between two observed point clouds, while motion prediction methods aim to forecast future displacements based on past observations; (2) Estimating dense 3D flow is time-consuming, which is not feasible for self-driving systems [15]. Furthermore, directly applying scene flow estimation on real LiDAR point clouds poses challenges due to the lack of one-to-one correspondences [19, 30].

3. Method

In this section, we begin by introducing the problem formulation of class-agnostic motion prediction. Subsequently,

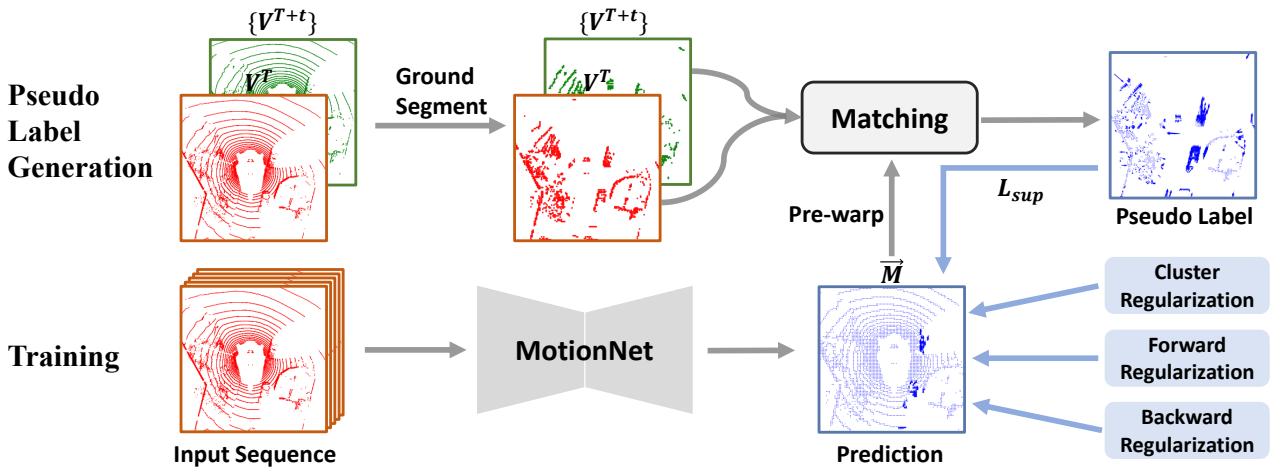


Figure 2. Overview of the proposed approach. Without ground truth labels, we first generate pseudo labels by matching. We then introduce cluster, forward, and backward regularization losses to facilitate self-supervised motion learning.

we delve into a comprehensive presentation and discussion of each component comprising our approach. The overview of our approach is illustrated in Fig. 2.

3.1. Problem Formulation

Given a temporal sequence of LiDAR point cloud frames, past frames are first synchronized to the current coordinate system [31]. We denote synchronized point cloud captured at time t as P^t. Following [31], we discretized P^t into dense voxels $V^t \in \{0,1\}^{H\times W\times C}$, where 0 indicates the voxel is empty, 1 indicates the voxel is occupied by at least 1 point, and H, W, C are the voxel numbers along X, Y, Z axis respectively. Then we represent the 3D voxel lattice as a 2D pseudo-image with the C corresponding to the image channel and view V^t as a virtual BEV map. Then the motion field $M \in \mathbb{R}^{H\times W\times 2}$ in the BEV map is defined as the 2D displacement of each grid cell to the future timestamp. Taking sequence $\overrightarrow{\mathcal{V}} = \{V^t\}_{t=1}^{T}$ as input, the task aims to predict the motion fields $\overrightarrow{\mathcal{M}} = \{M^{T\to T+t}\}_{t=1}^{T'}$, where T and T' are the number of past and future timestamps, respectively.

3.2. Overview

As depicted in Fig. 2, our approach mainly involves pseudo label generation and self-supervised training. To generate pseudo labels, we initially solve the matching problem between two consecutive frames. Firstly, we remove the majority of ground plane points from point clouds by ground segmentation algorithm [12] to reduce noise for better matching. Subsequently, we feed the raw BEV map sequence into MotionNet [31] and train the model using the generated pseudo labels through supervised loss $\mathcal{L}_{sup}$. As the quality of pseudo labels is low at the initial stage of training, we introduce three spatial and temporal consistency regularization losses, named cluster consistency loss $\mathcal{L}_c$, forward consistency loss $\mathcal{L}_f$, and backward consistency loss $\mathcal{L}_b$, to facilitate the self-supervised learning.

3.3. Pseudo Label Generation

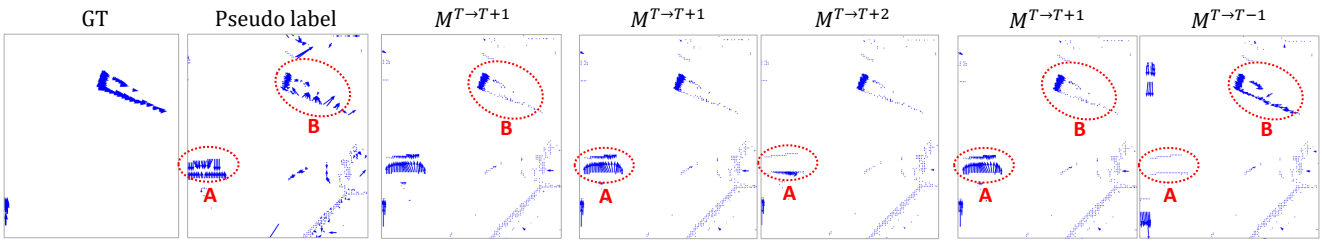
Unlike fully-supervised learning, where ground truth labels are available, self-supervised methods learn from data itself without annotations. In this condition, one alternative approach is to generate pseudo labels to provide the necessary supervision for training. Considering motion can be defined as the variation in coordinates between correspondent points in two consecutive frames, the process of generating pseudo labels can be viewed as a matching problem. We denote $B^T = \{b_i^T \in \mathbb{R}^2\}_{i=1}^{N_T}$ as the 2D coordinates of BEV cells on the current virtual BEV map V^T, where N_T is the number of non-empty cells. Then we perform matching between source B^T and target $B^{T+t} = \{b_j^{T+t} \in \mathbb{R}^2\}_{j=1}^{N_{T+t}}$. Ideally, if there exists one-to-one correspondence between B^T and B^{T+t}, B^T can be projected into B^{T+t} and fully occupy B^{T+t}:

$$B^T + M = \pi B^{T+t}, \tag{1}$$

where $\pi \in \{0,1\}^{N_T\times N_{T+t}}$ is a permutation matrix indicating correspondence between B^T and B^{T+t}, and M is the motion flow. The optimal soft permutation matrix $\pi^* \in [0,1]^{N_T\times N_{T+t}}$ can be computed by solving an optimal transport problem:

$$\begin{aligned} \pi^* &= \arg\min_{\pi} \sum_{i,j} C_{ij}\pi_{ij} \\ s.t. \quad \pi \mathbf{1}_{N_{T+t}} &= \frac{1}{N_T}\mathbf{1}_{N_T}, \pi^T \mathbf{1}_{N_T} = \frac{1}{N_{T+t}}\mathbf{1}_{N_{T+t}}, \end{aligned} \tag{2}$$

where $\mathbf{1}_{N_T} \in \mathbb{R}^{N_T}$ and $\mathbf{1}_{N_{T+t}} \in \mathbb{R}^{N_{T+t}}$ are vectors with all elements are 1. C is the transport cost matrix and π^T is the transpose of π. This problem can be solved by the Sinkhon



(a) GT & Pseudo label (b) Spatial incons. (c) Forward temporal incons. (d) Backward temporal incons.

Figure 3. Prediction results with inconsistency. (a) Ground truth and pseudo labels. (b) Predictions from the same object are inconsistent. (c) Predictions for consecutive future timestamps are inconsistent. (d) Forward and backward predictions (*e.g.*, $M^{T\to T+1}$ and $M^{T\to T-1}$) are inconsistent. Inconsistency regions are highlighted by red circles. The blue arrow denotes the future displacement (motion).

algorithm [4]. We use a pre-warping operation to warp the source BEV cells B^T by the predicted motion $M^{T\to T+t}$, and find the correspondence between the warped BEV cells $\hat{B}^T = B^T + M^{T\to T+t}$ and target B^{T+t} to calculate C. In this way, with better predictions, the quality of pseudo labels keeps improving during training. The cost matrix C is obtained by computing the pairwise distance between coordinates:

$$C_{ij} = 1 - \exp\left(-\frac{||\hat{b}_i^T - b_j^{T+t}||^2}{\theta_c}\right), \tag{3}$$

where θ_c is a temperature parameter. According to Eq.1, the pseudo motion labels can be obtained by:

$$\hat{M}^{T\to T+t} = \pi^* B^{T+t} - B^T. \tag{4}$$

With the pseudo motion labels, the supervised training loss can be computed by:

$$\mathcal{L}_{sup} = \sum_{t=1}^{T'} smooth_{L1}(M^{T\to T+t}, \hat{M}^{T\to T+t}) \tag{5}$$

During pseudo label generation, we first employ a training-free ground segmentation algorithm [12] to remove the majority of ground plane points to reduce the negative influence of ground points for matching and set the motion labels of these removed points as zeros. θ_c is set as 3 in the paper.

3.4. Spatial-Temporal Consistency Regularizations

3.4.1 Motivations

Due to factors such as occlusion, ground points, and sensors moving, establishing correct correspondences between source and target BEV maps is challenging. Despite removing ground points by the ground segmentation algorithm, noises without correspondences remain. The existence of these noises and cells without correspondences significantly impairs the solution to optimal transport problems. Consequently, the generated pseudo labels are inaccurate, leading to suboptimal outcomes.

For instance, certain cells corresponding to static background regions may get apparent moving labels (*e.g.*, circle A in Fig. 3 (a)), while cells associated with moving objects might possess static or inaccurate moving labels (*e.g.*, circle B in Fig. 3 (a)). Training the model only using such low-quality pseudo labels yields unsatisfactory results including the spatial and temporal inconsistent predictions, as shown in Fig. 3 (b)(c)(d). Therefore, we introduce three spatial and temporal consistency regularization terms. These regularization terms aim to penalize inaccurate predictions and propagate accurate ones. And through the pre-warping operation, the quality of the pseudo labels can improve iteratively as the accuracy of predictions increases.

3.4.2 Spatial cluster consistency regularization.

In real-world traffic scenarios, the majority of objects, such as cars and trucks, exhibit a high degree of rigidity. It is intuitive to expect that cells belonging to the same rigid object should display consistent predicted motions. Unfortunately, the generated pseudo labels for cells within the same rigid object are inconsistent (*e.g.*, circle B in Fig. 3 (a)). Consequently, directly learning from these pseudo labels may result in situations like half of the object being predicted with moving status but the other half being predicted with static status (*e.g.*, circle B in Fig. 3 (b)). Therefore, encouraging the consistency of predicted motion within the same object becomes an important factor in enhancing the performance of the self-supervised model.

Due to the unavailability of instance segmentation labels, one potential solution is to employ K-Nearest-Neighbors (KNN) consistency loss [32]. Although this loss helps encourage local smoothness, it does not consider instance-level consistency, especially in the case of large instances. To this end, we utilize cluster consistency loss to perform the spatial consistency constraint. As cells belonging to the same object are often in close proximity to one another in the virtual BEV map, objects can be coarsely represented as compact clusters. Therefore, we first utilize a breadth-first clustering algorithm to cluster every single cell within

a distance d_c (The detailed clustering process is in the Supplementary). Then for each cluster, we encourage spatial consistency. As illustrated in Sec. 3.3, we perform clustering on 2D BEV maps to reduce the computational cost. The cluster consistency loss is formulated as:

$$\mathcal{L}_c = \frac{1}{|S|}\sum_{s_i \in S}\frac{1}{|s_i|^2}\sum_{b_i, b_j \in s_i}\left(||\overrightarrow{\mathcal{M}}(b_i) - \overrightarrow{\mathcal{M}}(b_j)||\right), \quad (6)$$

where $S = \{s_i\}_{i=1}^{N_s}$, s_i indicates a cluster, N_s is the cluster numbers in the BEV map and $\overrightarrow{\mathcal{M}}(b)$ is the motion field at cell b. d_c is set as 3 in the paper.

3.4.3 Temporal forward consistency regularization.

From supervised loss $\mathcal{L}_{sup}$, the motions are learned individually from the pseudo labels for each timestamp. However, since the foreground and noise distributions of future BEV maps are different (*e.g.* B^{T+1} and B^{T+2}), we will get inconsistent pseudo motion labels between $\hat{M}^{T\to T+1}$ and $\hat{M}^{T\to T+2}$. As the model learns from these inconsistent pseudo labels, it will produce inconsistent predictions. For example, the predictions of static cells in $M^{T\to T+1}$ are wrong but correct in $M^{T\to T+2}$ (circle A in Fig. 3 (c)). As the object displacements in two consecutive short time windows are supposed to be consistent, we regularize forward predictions as:

$$\mathcal{L}_f = \sum_{t=1}^{T'-1} smooth_{L1}(M^{T\to T+t}, \frac{t}{t+1}M^{T\to T+t+1}). \quad (7)$$

With forward consistency loss $\mathcal{L}_f$, the predicted motions of adjacent timestamps act as mutual regularization, which is important for self-supervised motion learning as the pseudo labels are temporally inconsistent. For example, $M^{T\to T+2}$ can provide more accurate information for $M^{T\to T+1}$ in Fig. 3 (c) case.

3.4.4 Temporal backward consistency regularization.

When generating pseudo labels, it is inevitable to assign noise and some ground source cells to the wrong target cells due to the absence of corresponding cells in the target point clouds. Wrong pseudo labels consequently lead the trained model to predict incorrect motion, especially wrongly predicting large displacements for background cells (*e.g.* circle A in left Fig. 3 (d)). This creates a contradiction between the model's expected behavior (which is to predict static motion for background cells based on the past frames) and the actual predictions learned directly from incorrect pseudo labels. Inspired by the contradiction, we propose backward consistency loss to penalize incorrect predictions.

Specifically, besides forward sequence $\overrightarrow{\mathcal{V}} = \{V^1, V^2, ..., V^T\}$, we also feed additional backward sequence $\overleftarrow{\mathcal{V}} = \{V^{2T-1}, V^{2T-2}, ..., V^T\}$ to the network to predict the backward motion $\overleftarrow{\mathcal{M}}$. Compared to the forward sequence, the backward sequence has entirely different past frames. During training, without being directly constrained by pseudo labels like the forward motion, the backward motion reflects the model's general ability to perceive motion cues hidden in the temporal sequences. Considering that the forward and backward speeds of an object are opposites at the same moment, we formulate the backward regularization term as:

$$\mathcal{L}_b = \sum_{t=1}^{T'}\exp(-\frac{t}{\theta_b})smooth_{L1}(M^{T\to T+t}, -M^{T\to T-t}), \quad (8)$$

where θ_b is the temperature parameter. We use the $\exp(-\frac{t}{\theta_b})$ term to dynamically adjust the loss weights, taking into account that as the time horizon increases, accurately predicting motion becomes more challenging, and the forward-backward divergence becomes less reliable. This inconsistency becomes noticeable in cells with incorrect predictions (Fig. 4), which demonstrates that backward regularization can effectively reflect the accuracy of the predictions. Thus, the backward regularization $\mathcal{L}_b$ generally penalizes incorrect predictions while having less impact on correct ones. θ_b is set as 10 in the paper. In Sec. 4, we will show that the backward regularization effectively benefits the self-supervised motion learning process.

Overall loss functions. To summarize, the overall training objective is:

$$\mathcal{L}_{total} = \mathcal{L}_{sup} + \alpha\mathcal{L}_c + \beta\mathcal{L}_f + \gamma\mathcal{L}_b, \quad (9)$$

where α, β, and γ are hyper-parameters, balancing the contribution of the different loss terms. It is worth noting that all the regularization terms are only used for training.

4. Experiments

In this section, we begin by comparing our methods with state-of-the-art (SOTA) motion prediction methods. Subsequently, we conduct ablation studies to analyze the effectiveness of each individual component.

Dataset. We mainly evaluate the proposed approach on the large-scale self-driving dataset: nuScenes [2], which contains 850 scenes with annotations. For fair comparisons, we follow previous works [19, 30, 31] to use 500 scenes for training, 100 scenes for validation, and 250 scenes for testing.

Implementation details. We follow the same pre-process setting in [31], where the point clouds are cropped in the range of [-32m, 32m]× [-32m, 32m]× [-3m, 2m] and the voxel size is set to 0.25m ×0.25m × 0.4m along XYZ axis. We take the current sweep and the past four sweeps as input. We use MotionNet as the baseline model for its effectiveness and flexibility following [15, 19]. We train the

Table 1. Performance comparison on nuScenes dataset. Full., Weak., and Self. refer to fully-supervised, weakly-supervised, and self-supervised training, respectively. Our approach surpasses state-of-the-arts by a large margin.

Method	Sup.	Modality	Years	Static		Speed $\leq$ 5m/s (Slow)		Speed $\geq$ 5m/s (Fast)	
				Mean↓	Median↓	Mean↓	Median↓	Mean↓	Median↓
Static	-	LiDAR	-	0	0	0.6111	0.0971	8.6517	8.1412
FlowNet3D [18]	Full.	LiDAR	CVPR2018	0.0410	0	0.8183	0.1782	8.5261	8.0230
HPLFlowNet [8]	Full.	LiDAR	CVPR2019	0.0041	0.0002	0.4458	0.0960	4.3206	2.4881
LSTM-ED [25]	Full.	LiDAR	ICRA2019	0.0358	0	0.3551	0.1044	1.5885	1.0003
MotionNet [31]	Full.	LiDAR	CVPR2020	0.0256	0	0.2565	0.0962	1.0744	0.7332
BE-STI [30]	Full.	LiDAR	CVPR2022	0.0220	0	0.2115	0.0929	0.7511	0.5413
WeakMotion [15]	Weak.	LiDAR	CVPR2023	0.0558	0	0.4337	0.1305	1.7823	1.0887
PointPWC [32]	Self.	LiDAR	ECCV2020	0.5264	0	0.9423	0.3225	3.9530	2.5149
RigidFlow [14]	Self.	LiDAR	CVPR2022	0.6361	0	1.1836	0.5666	3.9439	2.5781
PillarMotion [19]	Self.	LiDAR+Image	CVPR2021	0.1620	0.0010	0.6972	0.1758	3.5504	2.0844
WeakMotion [15]	Self.	LiDAR	CVPR2023	0.1919	0	0.4510	0.1150	2.9715	1.8822
Ours	Self.	LiDAR	-	**0.0419**	**0**	**0.3213**	**0.1061**	**2.2943**	**1.0508**

model for 20 epochs and set the learning rate to be 0.002. Adam is used as the optimizer. We implement our model in Pytorch [22] with a single RTX 3090 GPU. Hyperparameters α, β, and γ are set as 0.05, 0.1, and 1 respectively. Following previous works, we take the sequence with 5 frames ($T = 5$) as input and show the results of the next 1s ($T' = 5$) in the following tables. The time interval between each frame is 0.2s.

Evaluation metrics. Following previous works [19, 30, 31], we evaluate the mean and median errors at different speed levels by dividing the grid cells into 3 groups according to the ground truth speeds: static, slow ($\leq$ 5m/s) and fast ($\geq$ 5m/s). Errors are measured by L_2 distances between the predicted displacements and the ground truth displacements.

4.1. Comparison with State-of-the-Art Methods

In Table 1, we show the results of the following methods: (1) A static model that assumes a static environment. (2) Scene flow estimation methods, including FlowNet3D [18], HPLFlowNet [8], PointPWC [32], and RigidFlow [14]. Following [31], we employ scene flow estimation methods by assuming linear dynamics. First, we predict the motion by estimating the flow from the current time to the past 0.4s. Then, we project the predicted flow onto the BEV map and to the future 1.0s for comparison. (3) Class-agnostic motion prediction methods, including LSTM-ED [25], MotionNet [31], PillarMotion [19], BE-STI [30], and WeakMotionNet [15].

Compared to fully-supervised models, our approach outperforms FlowNet3D and HPLFowNet at both slow and fast speed levels, and it surpasses the weakly-supervised WeakMotionNet (with 1% foreground and background masks [15]) at static and slow speed levels. In comparison with self-supervised methods, our approach achieves superior performance to PillarMotion, with significant improvements of 74.1%(-0.1201 m), 53.9% (-0.3759 m), and 35.4%(-1.2561 m) at static, slow, and fast speed levels, respectively. To further show the superiority of the proposed approach, we implement recent WeakMotionNet in a self-supervised manner by replacing its pre-segmentation network with the same ground segmentation algorithm [12] used in our approach. While self-supervised WeakMotionNet outperforms PillarMotion at slow and fast speed levels, our approach still surpasses it by a large margin.

4.2. Ablation Studies

In this subsection, we evaluate the effectiveness of each component of our approach. As shown in Table 2, the base model trained only with pseudo labels does not perform well at all speed levels (Row 1).

Effectiveness of cluster consistency regularization. When cluster consistency loss is not applied, the model exhibits noticeably poorer performance at the fast speed level (Compare Row 1 and Row 6). This disparity in performance can be attributed to the way the cost matrix for the optimal transport problem is computed based on the distance between cells. Consequently, the solver tends to find correspondences between cells that are in close proximity to each other. While this proximity-based approach has its merits, it presents challenges in generating long displacement pseudo labels that the model can learn from. As a result, during the early stages of training, there is a scarcity of correct pseudo labels for fast motion. By imposing constraints on cluster consistency, the model is encouraged to predict faster motion, even if it corresponds to a short displacement pseudo label. This is possible due to the presence of fast displacement predictions within the same cluster. With larger displacement and pre-warping, the optimal transport solver is more likely to find correct correspondence between two

Table 2. Ablation study for different combinations of proposed regularization terms. Bold fonts and underlines indicate the best and the second-best performance, respectively.

#Row	$\mathcal{L}_{sup}$	$\mathcal{L}_c$	$\mathcal{L}_f$	$\mathcal{L}_b$	Static	Slow	Fast
					Mean Error ↓		
1	✓				0.1650	0.5540	3.4503
2	✓			✓	**0.0413**	<u>0.3249</u>	2.7587
3	✓		✓		0.1197	0.4264	3.2072
4	✓		✓	✓	0.0499	0.3277	2.5753
5	✓	✓		✓	0.0502	0.3432	<u>2.5154</u>
6	✓	✓			0.1705	0.4154	2.7027
7	✓	✓	✓		0.1450	0.4087	2.6509
8	✓	✓	✓	✓	<u>0.0419</u>	**0.3213**	**2.2943**

points at long distances. In Table 3, we present the results of KNN consistency loss and cluster consistency loss, both of which function as spatial smooth losses. Compared to $\mathcal{L}_{KNN}$, $\mathcal{L}_c$ bring more benefits comprehensively.

Effectiveness of temporal consistency regularization. Both forward and backward regularization terms boost the performances across all speed levels (Compare Row 5 and Row 7 with Row 8 respectively). To further illustrate the impact of backward regularization, we separately feed forward and backward sequences of all training samples into the model only trained with $\mathcal{L}_{sup}$ to demonstrate prediction divergence ($|\overrightarrow{\mathcal{M}} - \overleftarrow{\mathcal{M}}|$). As depicted in Fig. 4, there is a positive correlation between the forward-backward prediction divergence and prediction accuracy. The divergence is large when the prediction is inaccurate and vice versa, which indicates that backward regularization will have a stronger impact on inaccurate predictions but a weaker influence on accurate ones. By using the forward-backward divergence as a loss term, the model is penalized for making inaccurate predictions. Results for various θ_b are shown in Table 4, which indicates that the weight term can further improve the performance.

Qualitative results. We present the result visualizations of our approach in Fig. 5 to qualitatively show the impacts of each regularization term. When directly training with pseudo labels $\mathcal{L}_{sup}$, the model tends to produce inconsistent predictions for the same object and predict large motions for background cells. The third column illustrates that cluster consistency loss $\mathcal{L}_c$ effectively benefits spatial consistency. With the addition of forward regularization $\mathcal{L}_f$, the prediction results are further improved, but there are still noticeable noises in the background cells. Finally, by incorporating backward regularization $\mathcal{L}_b$, the noise predictions in the background cells are significantly suppressed.

Table 3. Impact of different spatial regularizations. Cluster-level regularization performs better than the KNN-based regularization. Bold fonts and underlines indicate the best and the second-best performance, respectively.

Method		Static	Slow	Fast
		Mean Errors ↓		
Baseline		0.0502	0.3432	2.5154
+ $\mathcal{L}_{KNN}$	$K = 3$	**0.0371**	0.3370	2.5362
	$K = 5$	0.0490	0.3378	2.4568
	$K = 10$	0.0731	0.3405	2.5116
+ $\mathcal{L}_c$ (ours)	$d_c = 2$	0.0526	0.3438	**2.1814**
	$d_c = 3$	<u>0.0419</u>	**0.3213**	<u>2.2943</u>
	$d_c = 4$	0.0459	<u>0.3318</u>	2.4434

Table 4. Varying θ_b for backward consistency loss. - indicates the results without the $\exp(-\frac{t}{\theta_b})$ term. Bold fonts and underlines indicate the best and the second-best performance, respectively.

θ_b	Static	Slow	Fast
	Mean Errors ↓		
5	0.0482	0.3264	<u>2.3580</u>
10	0.0419	<u>0.3213</u>	**2.2943**
15	**0.0293**	**0.3091**	2.4149
–	<u>0.0382</u>	0.3298	2.4596

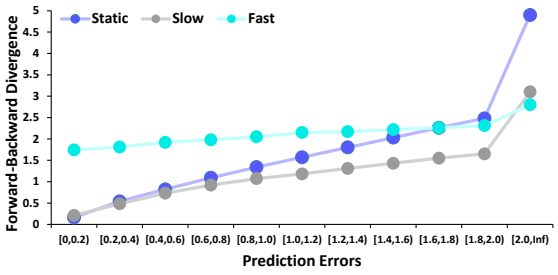


Figure 4. Forward-backward divergence of the training set. We analyze the prediction errors of all samples in the training set and their corresponding forward-backward divergences. The Forward-backward divergence is positively correlated with the prediction error.

4.3. Results on Waymo Dataset

Following WeakMotionNet [15], we further evaluate our approach on the Waymo dataset [27]. As shown in Table 5, similar to the results on the nuScenes dataset, our approach achieves mean errors of around 0.05m, 0.4m, and 2.3m at static, slow, and fast speed levels, respectively, outperforming self-supervised WeakMotionNet.

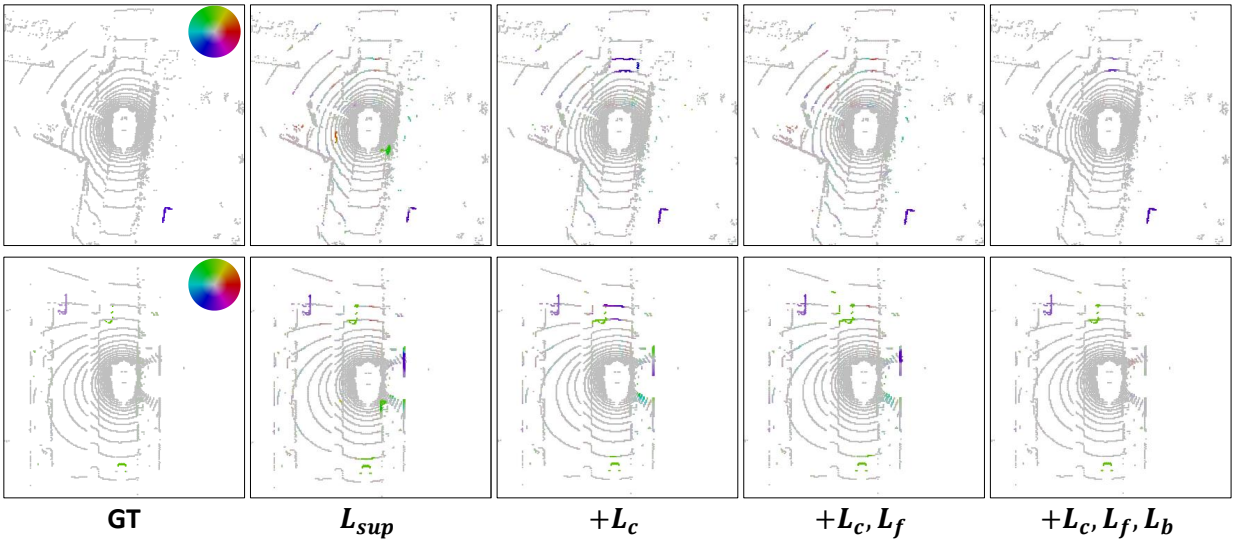


Figure 5. Qualitative results of the proposed self-supervised approach. The future displacements ($M^{T \to T+5}$) are depicted using the color wheel representation. (Zoom in for the best view)

Table 5. Motion prediction results on the Waymo Dataset. Our method still outperforms self-supervised WeakMotion by a large margin.

Method	Sup.	Static	Slow	Fast
		Mean Errors ↓		
MotionNet [31]	Full.	0.0263	0.2620	0.9493
WeakMotion [15]	Weak.	0.0297	0.3458	1.5655
WeakMotion [15]	Self.	0.1345	0.5833	3.0180
Ours	Self.	**0.0515**	**0.4440**	**2.3692**

4.4. Discussions

Ground Segmentation. We evaluate the performance of the ground segmentation algorithm [12] on the nuScenes dataset, and get a 95% precision rate while a 92% recall rate, indicating the majority of ground points can be correctly removed. Although almost all ground points can be successfully removed, the self-supervised training process still suffers from remaining ground points, noises, and points without correspondences. The issues are alleviated using our proposed regularizations, as shown in Fig. 2.

Clustering. In the paper, we utilize the breadth-first clustering (BFS) algorithm to demonstrate the effectiveness of the cluster consistency loss. We evaluate the clustering results of BEV cells on the nuScenes dataset and observe a 9.6% under-clustering rate and an 11.1% over-clustering rate. It indicates that although effective, the BFS algorithm still wrongly clusters cells from different instances in some cases. Therefore, fellow researchers can consider utilizing or developing more advanced clustering strategies, which may lead to better performances.

Efficiency. During training, ground segmentation and clustering take around 5ms and 10ms, respectively. In practice, these processes can be employed as data pre-processing before training. For the optimal transport (OT) solver, we implement it with Pytorch and operate in batch. It takes less than 1ms to solve the OT problem. During inference, given that our approach is model-independent, the inference time is based on the model used. When applying MotionNet, it takes 10ms for point cloud transformation and voxelization and another 10ms for prediction [31].

5. Conclusion

In this paper, we introduce a novel approach for self-supervised motion prediction with only unlabeled LiDAR point clouds. We utilize optimal transport to generate pseudo labels and propose three novel spatial and temporal regularization terms to facilitate self-supervised motion learning. Extensive experiments on the large-scale nuScenes dataset illustrate that our approach significantly surpasses previous self-supervised methods and shallows the gap with fully-supervised ones.

Acknowledgements. This study is supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s). This research is also supported by the MoE AcRF Tier 2 grant (MOE-T2EP20220-0007) and the MoE AcRF Tier 1 grant (RG14/22).

References

[1] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[2] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5

[3] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[4] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Neural Information Processing Systems (NeurIPS)*, 2013. 4

[5] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, Nitin Singh, and Jeff Schneider. Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020. 1, 2

[6] Guanting Dong, Yueyi Zhang, Hanlin Li, Xiaoyan Sun, and Zhiwei Xiong. Exploiting rigidity constraints for lidar scene flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[7] Liangji Fang, Qinhong Jiang, Jianping Shi, and Bolei Zhou. Tpnet: Trajectory proposal network for motion prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2

[8] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 6

[9] Marcel Van Herk. Image registration using chamfer matching. *Handbook of Medical Imaging*, 2000. 2

[10] Yair Kittenplon, Yonina C Eldar, and Dan Raviv. Flowstep3d: Model unrolling for self-supervised scene flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[11] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[12] Seungjae Lee, Hyungtae Lim, and Hyun Myung. Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3d point cloud. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022. 3, 4, 6, 8

[13] Ruibo Li, Guosheng Lin, and Lihua Xie. Self-point-flow: Self-supervised scene flow estimation from point clouds with optimal transport and random walk. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[14] Ruibo Li, Chi Zhang, Guosheng Lin, Zhe Wang, and Chunhua Shen. Rigidflow: Self-supervised scene flow learning on point clouds by local rigidity prior. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 6

[15] Ruibo Li, Hanyu Shi, Ziang Fu, Zhe Wang, and Guosheng Lin. Weakly supervised class-agnostic motion prediction for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2, 5, 6, 7, 8

[16] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. *Neural Information Processing Systems (NeurIPS)*, 2021. 2

[17] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2

[18] Xingyu Liu, C. Qi, and Leonidas J. Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 6

[19] Chenxu Luo, Xiaodong Yang, and Alan Loddon Yuille. Self-supervised pillar motion learning for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 5, 6

[20] Wenjie Luo, Binh Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[21] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems (NeurIPS)*, 2019. 6

[23] Jhony Kaesemodel Pontes, James Hays, and Simon Lucey. Scene flow from point clouds with or without learning. In *International Conference on 3D Vision (3DV)*, 2020. 2

[24] Gilles Puy, Alexandre Boulch, and Renaud Marlet. Flot: Scene flow on point clouds guided by optimal transport. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[25] Marcel Schreiber, Stefan Hoermann, and Klaus Dietmayer. Long-term occupancy grid prediction using recurrent neural networks. In *IEEE international conference on robotics and automation (ICRA)*, 2019. 1, 2, 6

[26] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang chun Woo. Convolutional lstm network: A machine learning approach for precipita-

tion nowcasting. In *Neural Information Processing Systems (NeurIPS)*, 2015. 2

[27] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 7

[28] Guangming Wang, Xinrui Wu, Zhe Liu, and Hesheng Wang. Hierarchical attention learning of scene flow in 3d point clouds. *IEEE Transactions on Image Processing (TIP)*, 2021. 2

[29] Kewei Wang, Yizheng Wu, Zhiyu Pan, Xingyi Li, Ke Xian, Zhe Wang, Zhiguo Cao, and Guosheng Lin. Semi-supervised class-agnostic motion prediction with pseudo label regeneration and bevmix, 2023. 1

[30] Yunlong Wang, Hongyu Pan, Jun Zhu, Yu-Huan Wu, Xin Zhan, Kun Jiang, and Diange Yang. Be-sti: Spatial-temporal integrated network for class-agnostic motion prediction with bidirectional enhancement. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 5, 6

[31] Pengxiang Wu, Siheng Chen, and Dimitris N. Metaxas. Motionnet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 3, 5, 6, 8

[32] 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 (ECCV)*, 2020. 2, 4, 6

[33] Maosheng Ye, Jiamiao Xu, Xunnong Xu, Tengfei Wang, Tongyi Cao, and Qifeng Chen. Bootstrap motion forecasting with self-consistent constraints. In *IEEE Conference on International Conference on Computer Vision (ICCV)*, 2023. 2

[34] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, Congcong Li, and Dragomir Anguelov. Tnt: Target-driven trajectory prediction. In *Conference on Robot Learning (CoRL)*, 2020. 1, 2

[35] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2