# Point Cloud Densification based on Scene Flow Estimation and Kalman Refinement

Yufei Que, Junzhe Ding, Jie Xie, Jin Zhang, Luqin Ye and Cheng Wu

***Abstract*— Point cloud densification is an effective measure to alleviate the sparseness of point clouds. In 3D vision, the positional relationship of multi-frame point clouds is applied to point cloud densification research to explain the rationality of the source of supplementary points. Among them, scene flow estimation is effective for dynamic scenes. However, scene flow estimation of long-sequence dynamic point clouds is prone to cumulative positioning errors. In order to solve this problem, this paper proposes to correct the scene flow estimation results from a timing perspective based on Kalman filtering. Specifically, the scene flow estimation model is first optimized according to the pyramid structure to improve the reliability of point cloud feature extraction. Then, combined with the temporal relationship of the point clouds in the previous and later frames, the point cloud is reconstructed uniformly to complete the densification of the point cloud. Finally, the densified point cloud is applied to the 3D detection task. Results on the KITTI3D tracking dataset show that the point cloud densification method based on scene flow estimation can effectively improve the performance of LiDAR-only detectors.**

***Index Terms*— point cloud densification, scene flow estimation, Kalman filter**

## I. Introduction

COMPARED with two-dimensional data, three-dimensional point clouds provide additional depth information. This makes point clouds widely used in assisted driving [1], robotics [2] and other fields. However, limitations such as object occlusion and scanning environment inevitably lead to point clouds being sparse or even partially missing. The sparse and discrete characteristics of point clouds dismatch the requirements for high-precision perception. Point cloud densification is one of the most effective measures to promote the accuracy of three-dimensional sensing. Therefore, obtaining high-quality dense complete point clouds is an urgent task.

The point cloud densification task aims to process sparse, uneven point clouds into dense, uniform point clouds. Analogous to an image super-resolution task, some studies also call this task a point cloud super-resolution (PSR) task. The high-resolution (HR) output is generated from a low-resolution (LR) input. Such tasks are generally divided into two methods: generating virtual points for single-frame point clouds and registering multi-frame point clouds to merge into a frame.

The most commonly used point cloud densification method is to generate virtual points based on point cloud coordinates. Inspired by PointNet++ [3], PU-Net proposed a point cloud upsampling model based on deep learning for the first time [4]. In order to prevent the upsampled points from clustering around the original points, repulsion loss and reconstruction loss are also introduced to help the network perform end-to-end training. However, this method only targets a single point, ignores neighborhood information, and easily loses local fine-grained details. In order to solve this problem, EC-Net maps the neighborhood information around each point into a feature vector [5]. Then, it uses a feature expansion strategy similar to PU-Net to learn the perturbation of the generated point cloud relative to the original point cloud position to obtain distance characteristics. However, the points generated by such methods are virtual points, and it is usually difficult to explain the rationality of the source of the points. Therefore, the multi-frame point cloud registration method is applied to point cloud densification.

The registration method [6]–[10] obtains a dense point cloud map by aligning multi-frame point clouds. Commonly used point cloud registration methods include random sampling consistency (RANSAC), normal distribution transformation (NDT) and Iterative closest point (ICP). By introducing global structural information in point clouds, GeoTransformer can significantly improve the inlier ratio of correspondences and achieve RANSAC-free scene point cloud registration [9]. RoReg uses the estimated local rotation of a matching point pair to directly find the global rigid transformation along with the translation of that point pair [10]. This enables RoReg to generate transformation hypotheses with only one correspondence, reducing the search space for transformations. Moreover, ICP is a classic registration method. It is based on the mathematical idea of the least squares method, which optimizes the correspondence between two point clouds by minimizing the difference between them. Although the ICP series of algorithms are designed to effectively align large-scale point clouds, they often ignore the dynamic properties of objects within the scene. With the emergence of deep learning, various techniques [11], [12] exploit feature correspondences and spatial relationships within point clouds to advance odometry methods. Scene flow estimation aims to understand the dynamic changes of the environment to determine the connection

The work is supported by the National Natural Science Foundation of China (No. 62275186 and 62201372). (Yufei Que and Junzhe Ding are co-first authors. Corresponding authors: Jin Zhang and Cheng Wu.)

Yufei Que, Junzhe Ding, Jie Xie, Jin Zhang, Luqin Ye, and Cheng Wu are with the School of Rail Transportation, Soochow University, Suzhou, 215500, China (e-mail: 20225246039@stu.suda.edu.cn; jzding@stu.suda.edu.cn; xiejie@suda.edu.cn; zhangjin1983@suda.edu.cn; lqye@suda.edu.cn; cwu@suda.edu.cn).

between the point clouds of previous and subsequent frames. Early works [13]–[16] focused on stereo vision cameras, where knowledge-driven methods were used to estimate 3D motion fields. Flownet3D [17] is the first end-to-end scene flow estimation network based on deep learning. It utilizes PointNet++ [3] as the backbone network and introduces a novel stream embedding layer. NSFP [18] is the first scene flow estimation method based on runtime optimization. It uses a unique implicit regularizer to represent scene flow and is not subject to data-driven constraints. However, when estimating long sequence point cloud scene flows, the estimator is limited by the motion relationship between the two frame point clouds, which may cause the point cloud to be guided to the wrong location.

This paper proposes a point cloud densification method based on scene flow estimation, aiming to increase the density of dynamic objects in long sequences of point clouds to enhance the performance of 3D object detection. In order to overcome the positioning error of scene flow estimation in long sequence point clouds, this paper splices a correction module based on Kalman filter to eliminate positioning errors frame by frame. Specifically, we aggregate multiple frame point clouds into the same frame via modified scene flow estimation. Then, for the merged object point cloud, uniform point cloud data is generated through reconstruction method. Our contributions to this study are as follows:

- We propose a novel method for densifying point clouds by utilizing motion information from multi-frame point clouds with a novel scene flow estimator using a two-branch pyramid network as the implicit regularizer.
- We address the issue of localization error in long-sequence scene flow estimation by utilizing a Kalman filter to precisely adjust the position of dynamic objects.
- We compare our point cloud densification method with the ICP-based point cloud densification method and our method outperforms in terms of accuracy and precision.
- We conduct extended experiments on the KITTI 3D tracking dataset to demonstrate that our method effectively improves the LiDAR-only detectors' performance by achieving superior results compared to the baselines.

## II. Related Work

### A. Spatial Registration for Point Clouds

ICP is the most commonly used point cloud registration algorithm. However, it will easily fall into local optimality and no clear range can define whether the ICP algorithm will fall into a local optimum. Consequently, [19] assesses the effectiveness of both point-to-point and point-to-plane ICP algorithms using overlap, angle, distance, and noise. GO-ICP [20] optimizes globally by combining the local ICP algorithm with the global boundary determination. Although these algorithms have overcome the limitations of traditional ICP The algorithm has shortcomings, but it has problems with poor robustness and low universality when processing a large number of point clouds. Consequently, [21] applied a k-dimensional Tree (KD-tree) improved ICP algorithm on the basis of coarse registration. [22] uses local optical flow for point matching, avoiding the expensive point matching process and thus accelerating the optimization step. [23] uses multi-resolution based on hierarchical octrees to improve calculation speed while overcoming the alignment risk caused by point density which is usually sparse and uneven. Although this algorithm has high registration accuracy and efficiency but the registration effect is not good enough for objects with more complex structures.

Considering the advantages and disadvantages of coarse registration and precise registration, the two-step registration algorithm has become mainstream. 3D normal distribution transform (3D-NDT) combined with ICP can effectively reduce registration time and ensure accuracy even when the amount of point cloud data is large [24]. Inproved 3D NDT-ICP introduces kd-tree into the ICP algorithm for point pair search, and uses the Gauss-Newton method to optimize the algorithm to solve the nonlinear objective function to complete the accurate registration of the tunnel tunnel point cloud [25]. But in general, ICP is more suitable for multi-pose spatial registration. With the increase in the availability of point cloud data, research on scene flow and related applications has gradually become a hot spot in the field of 3D environment sensing.

### B. Timing Registration for Point Clouds

Scene flow prediction is a type of registration task that registers two point clouds in a time sequence. With the emergence of point cloud feature encoding networks such as PointNet [26], PointNet++ [3] and DGCNN [27], scene flow prediction networks based on optical flow prediction emerged. FlowNet3D introduces Flow Embedding Layer and Set Upconv Layer to estimate scene flow from continuous point clouds in an end-to-end manner [17]. FlowNet3D++ overcomes the shortcomings of insufficient utilization of motion information in single-frame point clouds in FlowNet3D, and combines multi-frame information through MeteorNet to more fully predict displacements [28]. PointPWC-Net draws on the optical flow prediction network PWCnet, and for the first time introduces a coarse-to-fine method to process scene flow in 3D point clouds [29]. However, this method will lead to low-scale prediction errors, which can easily affect high-scale prediction results. Furthermore, since point clouds are irregular and disordered, it is challenging to efficiently extract features from the correlations between all point pairs in 3D space. Therefore, PV-RAFT proposes a method called Point-Voxel Correlation Fields [30]. This method captures the local and long-range dependencies of point pairs and effectively handles small and large displacements. Neural Scene Flow Prior (NSFP) introduces a neural scene flow prior, using the architecture of a neural network as a new implicit regularizer [18]. The neural prior implicit and continuous scene flow representation allows us to estimate dense long-term correspondences in point cloud sequences.

### C. Kalman Filter

Kalman filter has excellent performance in theory and practice [31], [32]. Kalman filtering uses a state space model

to represent the unobserved component model. This allows it to flexibly handle uncertainty and multiple state quantities. Furthermore, based on a dynamic model of the system, control inputs and multiple measurement sequences are considered. This makes Kalman very useful in sensor fusion and data fusion. Therefore, the Kalman filter is simple and effective in processing long sequence point clouds. In four-dimensional terrain point clouds, Kalman filtering combines the M3C2 distance with the uncertainty obtained through error propagation to temporally resample the estimated change values to improve modeling accuracy [33]. Kalman filtering is combined with SLAM to enable the robot to determine its position and analyze its surrounding environment [34].

## III. Proposed Method

In this paper, we propose a point cloud scene flow estimation method based on Kalman correction to reduce positioning errors. Based on the results of scene flow estimation, dynamic long-sequence point cloud densification is achieved. Specifically, this paper first estimates the scene flow based on a two-branch pyramid network to obtain the optimal estimation weight. In the process of using weights for long-sequence scene flow estimation, the Kalman filter is used to correct the estimated position of the dynamic target, thereby reducing the cumulative positioning error. Applying the above operations to the point cloud sequence from the initial frame to the final frame, the overall framework of the densification method is shown in Fig. 1.

### A. Long Sequence Scene Flow Estimation

In a three-dimensional scene, point cloud scene flow describes the motion of each point between consecutive frames, recovering the motion field composed of the motion vectors of each point in the given two or more point cloud scenes.

$P_t$ and $P_{t+1}$ are defined as the neighboring two-frame point clouds and $SF$ is defined as the scene flow. In the actual scene, there is no one-to-one correspondence between $P_t$ and $P_{t+1}$. Therefore, $SF$ represents the motion vector from each point in the $P_t$ point cloud to the most matching position in the $P_{t+1}$ point cloud.

$$SF_{t,t+1} = F\left(P_t, P_{t+1}, \theta\right) \tag{1}$$

where $F(.)$ denotes the scene flow estimation model and $\theta$ is the model weights. Scene flow estimation represents the dynamic relationship between paired point clouds of two frames in a point cloud sequence. For a given point cloud sequence $\{P_0,P_1,P_2,...,P_n\}$, the optimal scene flow model weights are $\{\theta_{0,1}, \theta_{1,2}, ..., \theta_{n-1,n}\}$. The classical forward Euler method is applied to estimate the scene flow sequentially.

$$SF_{i,j} = SF_{i,j-1} + F\left(P_{j-1}, P_j, \theta_{j-1,j}\right) \tag{2}$$

where $i$ and $j$ any two frames in the sequence.

### B. A Two-branch Pyramid Network

A two-branch pyramid network is designed to estimate the scene flow in Fig. 2. Specifically, it mainly includes three modules: pyramid feature extraction, MLP-based implicit regularization and regularizer constraints. The regularizer constraints are formulated into two parts which enable the output to satisfy the position and flow consistency.

In pyramid feature extraction module, a point cloud frame in the sequence $P_t$ is fed into a pyramid feature extractor. The pyramid consists of bottom-up and top-down pathways for the point cloud. The number of sampling layers is set to 4 for both the bottom-up and top-down pathways, and the scale rate is set to 2. The output of each downsampled layer will be merged with the results of the upsampling layer and fed into the next layer for the next upsampling operation. The final output, noted as $P_t^{pyr}$, is fed into an implicit regularizion module for optimization.

In the implicit regularizion module, we construct the backbone using an MLP-based neural network. We include a position head and a flow head to encode the positional and motion information of the input point cloud. We adopt the LeakyReLU [35] as the activation function. The input layer for the backbone has 3 channels, and there are 6 hidden layers with 128 hidden units. The backbone has a 128-channel output which is followed by the position head and flow head. Each head has two hidden layers with 128 hidden units. Each head has a 3-channel output, as it represents the estimated coordinates and motion vectors in the XYZ coordinate system. Given the $P_t^{pyr}$, the position head outputs the estimated coordinates noted as $P_{t+1}^{'}$, and the flow head outputs the estimated flow vectors noted as $SF_{t,t+1}$.

In the regularizer constraints, we formulate the constraints as two parts. On the one hand, the $P_t^{pyr}$ combined the $SF_{t,t+1}$ should be consistent with the $P_{t+1}$. On the other hand, the $P_{t+1}^{'}$ substract by the $SF_{t,t+1}$ should be consistent with $P_t$. Thus, the objective function can be defined as Equation 3.

$$\begin{aligned}\mathcal{F}\left(P_t^{pyr}, P_t, P_{t+1}, \theta\right) =& \Psi\left(P_t^{pyr} + SF_{t,t+1}, P_{t+1}\right) + \\ & \Psi\left(P_{t+1}^{'} - SF_{t,t+1}, P_t\right),\end{aligned} \tag{3}$$

where $\Psi(.,.)$ denotes the Chamfer distance function [36], which is defined as Equation 4.

$$\Psi(a, B) = \min_{b \in B} \|a - b\|_2^2, \tag{4}$$

where $a$ and $b$ are point from point cloud $A$ and $B$.

During the optimization process, we follow the gradient descent technique and obtain the optimal scene flow model weights $\theta^*$, as shown in Equation 5.

$$\theta^* = \arg\min_{\theta} \mathcal{F}\left(P_t^{pyr}, P_t, P_{t+1}, \theta\right) \tag{5}$$

### C. Kalman Filter Refinement

However, although all scene flow model weights$\{\theta_{0,1}, \theta_{1,2}, ..., \theta_{n-1,n}\}$are the optimal estimation of the corresponding point cloud pairs, these parameters tend to suffer from the problem of accumulating localization errors when performing

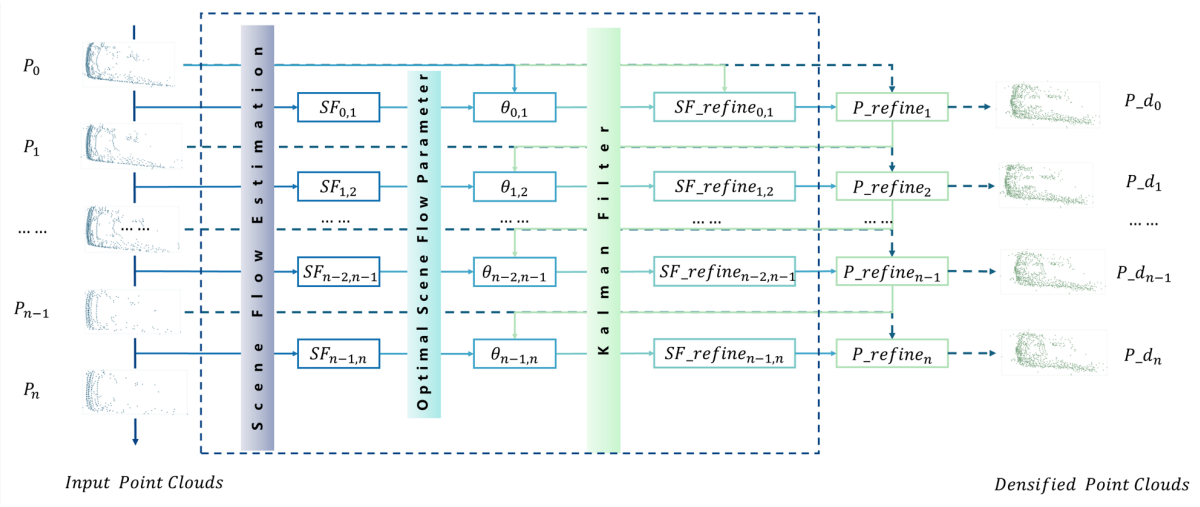


Fig. 1. The framework of densification method.

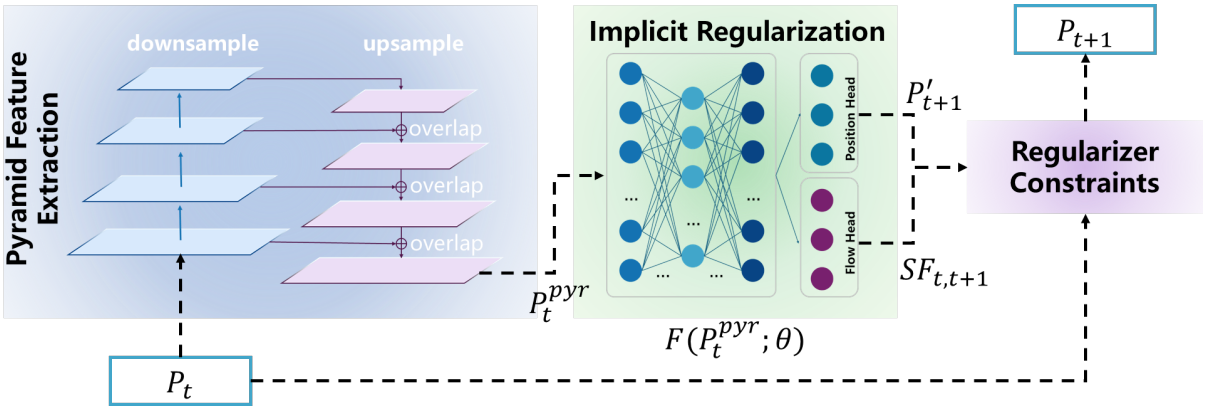


Fig. 2. Overall architecture of the two-branch pyramid network.

long sequences of scene flow estimation. To eliminate the cumulative localization error of point cloud during the long sequence estimation process, the Kalman filter [37] is introduced to correct the dynamic targets' position.

The cluster center of dynamic target is obtained as the state quantity $X = [x, y, z, vx, vy, vz]^T$ and assumes that the dynamic target conforms to the uniform motion, so the transfer matrix $A$ is set as

$$\begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

$X_{k-1|k-1}$ is the optimal outcome of the previous state, $A$ and $B$ are the system parameters, $U_k$ is the control quantity of the present state, and $X_{k|k-1}$ is the prediction of $X_{k-1|k-1}$. $P_{k-1|k-1}$ is the covariance of $X_{k-1|k-1}$, $A^{'}$ is the transpose matrix of $A$, $Q$ is the covariance of the system process noise, and $P_{k|k-1}$ is the covariance of $X_{k|k-1}$. $H$ is the observation matrix, $R$ is the noise covariance of the system measurements, and $Kg_k$ is the Kalman gain. $Z_k$ is the measured value, $X_{k|k}$ is the optimal predicted value at the current moment, and $P_{k|k}$ is the covariance of $X_{k|k}$.

The center of dynamic target is corrected using the optimal Kalman filter prediction value $X_{k|k}$, which eliminates the localization error frame by frame and obtains accurate scene flow estimation results for a long sequence of point clouds.

### D. Point Cloud Refinement

After accumulating multi-frame point clouds, duplicate points exist in the same part of the target. We propose refining and deduplicating the point cloud to reduce the noise accumulation on the densification result and ensure the uniformity of the target object's density. The average point cloud distance $D$ of point cloud $P_j$ is used as the search radius, and the average point cloud distance $D$ is calculated by Equation 6.

$$D = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2 + (z_i - \bar{z})^2}, \quad (6)$$

where $\bar{x}$,$\bar{y}$,$\bar{z}$ are the mean values of point cloud in $x$,$y$,$z$ dimensions respectively and $n$ is the number of point clouds.

For any point $p$ in the scene flow estimation result $P_{i,j}$ corrected by the Kalman filter, if a point in $P_j$ falls within the search radius of $p$, then $p$ is replaced with the nearest point in $P_j$; otherwise the point $p$ is retained. When the value of $i$ is taken all over the range from 0 to $j-1$, the densification results for the first $j$ frames are obtained.

## IV. Experiments

The performance of the proposed method is tested on KITTI tracking dataset [38]. The superiority of ICP in point cloud registration has been verified. This section introduces the experimental setup and then presents the results of densification. We first describe the settings of the scene flow estimation model, and then we compare the proposed method with the ICP algorithm on the densification results. For further analysis, five different LiDAR detectors are trained on the KITTI tracking datasets for evaluating the performance of ICP and our method.

### A. Experimental Setup

Table I gives the hardware and software configuration. Adam optimizer is used to minimize the objective function. The learning rate is set to 8e-3 and the optimization rounds to 500 iterations to avoid overfitting. Chamfer Distance (CD) [36] and Root Mean Square Error (RMSE) are two metrics.

TABLE I
HARDWARE AND SOFTWARE CONFIGURATION

| Configuration | Hardware/Software |
|---|---|
| System | Ubuntu20.04 |
| Environment | Pytorch1.12.0+CUDA11.6 |
| CPU | Intel i7-12700H@ |
| GPU | NVIDIA RTX 3080 Ti |

### B. Densification Results and Analysis

The densification experiments are carried out on cars, cyclists, and trucks. We employ random cropping to disrupt the structure of the raw point cloud. The experimental results of our method and the ICP algorithm are depicted in Fig.3. Based on the analysis of the experimental results, it is evident that our method effectively preserves the shape of the densified point cloud while minimizing the presence of noise points. A comparative analysis of the method's performance across various object classes is presented in Table II. As CD measures the consistency of the densification outcome and the ground truth and RMSE is used to weigh the accuracy of the densification outcome in relation to the corresponding actual values of the inner points, Table II demonstrates that the method proposed in this paper achieves lower results for all three classes in both metrics. The proposed method exhibits superior performance compared to the conventional ICP algorithm, as it effectively enhances the density and completeness of the point cloud.

TABLE II
COMPARISON OF POINT CLOUD DENSIFICATION PERFORMANCE

| | Trunk1 | Trunk2 | Car1 | Car2 | Cyclist1 | Cyclist2 |
|---|---|---|---|---|---|---|
| OURS-CD | **0.207862** | **0.277185** | **0.228993** | **0.206742** | **0.155147** | **0.099149** |
| ICP-CD | 0.250756 | 0.528042 | 0.321194 | 0.295102 | 0.172999 | 0.134865 |
| OURS-RMSE | **0.051484** | **0.061112** | **0.064533** | **0.048028** | **0.059912** | **0.050888** |
| ICP-RMSE | 0.063821 | 0.070297 | 0.069794 | 0.065152 | 0.068576 | 0.059299 |

### C. Densified point cloud four-channel detection

To obtain benchmarks for comparison experiments, we conducted training sessions for five chosen detectors using the raw KITTI tracking dataset. The dataset is shuffled and divided into $train$ (4001 frames) and $val$ (3900 frames) splits for evaluation purposes. Then, we densified the raw KITTI tracking dataset using one and three point cloud frames and investigated the impact of the proposed densification method on five different LiDAR detectors. The training setup is consistent with the Openpcdet framework [39]. The training epoch is set to 200 to converge the detectors. Table III illustrates that the proposed method led to significant improvements in PointPillars [40], SECOND [41], PointRCNN [42], PV-RCNN [43], and Part-$A^2$ [44]. Especially in hard situations, our proposed method offers up to 7.95% improvement in the cyclist category when using the PV-RCNN detector. It should be noted that these five LiDAR detectors show limited improvement on the three-frame densified dataset compared to the one-frame densified dataset. This suggests that the number of point cloud frames used for densification needs to be carefully determined under different situations.

The qualitative comparison of 3D object detection is depicted in Fig.5. The left column of images shows the results from the PV-RCNN detector trained on the three-frame densified dataset, while the right column of images shows the results from the PV-RCNN detector trained on the raw dataset. Our method provides additional structural and semantic information for the detector, which can help reduce false positive detections in general scenes. As illustrated in the highlighted section of the image, our approach successfully reduces the occurrence of false positive detections for vehicles, pedestrians, and cyclists.

### D. Densified point cloud multi-channel detection

In the fields of image processing and computer vision, the extraction of semantic tags is a key step in understanding and parsing image content. This paper carries out a three-dimensional detection task based on the semantic labels of densified point cloud hybrid images to verify the effect of densified point cloud. We first use the SLIC algorithm to segment the image into superpixels of similar size by iteratively optimizing the cluster center in color and spatial proximity. The goal is to divide the image into regions of consistent color and texture. The pre-trained deep learning model GMA is utilized to extract dense optical flow between pairs of images. The superpixel map generated by the SLIC algorithm is combined with the optical flow map obtained by the GMA model, and ISODATA clustering is used to group superpixels with similar motions. At this point, the mixed image semantic tags are obtained, such in Fig. 4. Combining densified point clouds and mixed image semantic labels to construct multi-channel point cloud data. Table IV illustrates the performance in PointPillars, SECOND, PointRCNN, PV-RCNN and Part-$A^2$.

## V. Conclusion

This paper proposes a Kalman-based scene flow estimation method for point cloud densification and 3D object detection

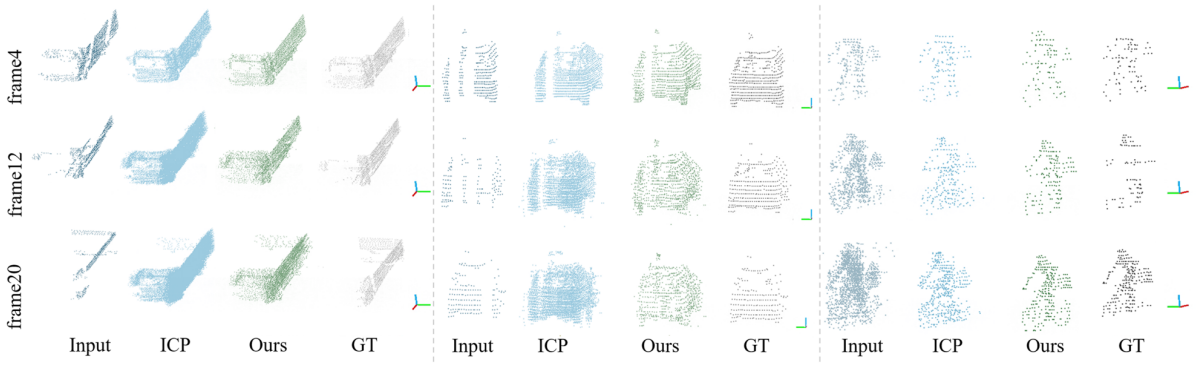


Fig. 3. Densified dynamic targets using 4, 12, and 20 point cloud frames.

TABLE III
COMPARISON OF FIVE DIFFERENT LIDAR DETECTORS' PERFORMANCE ON RAW AND DENSIFIED DATASET

| Method | Dataset | mAP | | Car | | Pedestrain | | Cyclist | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Hard | Easy | Mod. | Hard | Easy | Mod. | Hard | Easy | Mod. | Hard |
| PointPillars | Raw Dataset | 81.70 | 97.18 | 95.49 | 90.47 | 66.11 | 65.48 | 65.51 | 92.98 | 89.61 | 89.11 |
| | One-frame densified | **82.80** | **98.88** | **98.06** | **90.61** | **66.23** | **65.65** | **65.60** | **93.18** | **92.35** | **92.18** |
| | Three-frame densified | **86.06** | **98.83** | **98.51** | **96.15** | **67.76** | **67.50** | **67.30** | **98.06** | **95.78** | **94.73** |
| SECOND | Raw Dataset | 84.24 | **98.77** | 95.87 | 90.44 | 75.02 | 72.45 | 72.35 | 94.61 | **94.54** | 89.95 |
| | One-frame densified | **87.34** | 98.25 | **96.03** | **96.25** | **76.38** | **73.65** | **73.63** | **96.36** | 92.21 | **92.13** |
| | Three-frame densified | **87.64** | 98.21 | **97.42** | **96.15** | **76.15** | **74.28** | **74.69** | **94.78** | 92.11 | **92.09** |
| PointRCNN | Raw Dataset | 83.93 | 98.35 | 90.09 | 89.90 | 72.03 | 73.33 | 72.98 | 90.68 | 90.01 | 88.92 |
| | One-frame densified | **85.85** | 98.19 | **90.51** | **90.37** | **82.99** | **77.20** | **77.12** | **90.85** | **90.41** | **90.07** |
| | Three-frame densified | **86.16** | **98.95** | **90.55** | **90.39** | **80.76** | **79.35** | **77.13** | **96.91** | **91.37** | **90.97** |
| PV-RCNN | Raw Dataset | 89.59 | 99.49 | 96.70 | 96.82 | **82.34** | **82.05** | **81.91** | 98.16 | 96.17 | 90.03 |
| | One-frame densified | **91.54** | **99.50** | **97.33** | **96.89** | 81.20 | 80.85 | 81.54 | **98.78** | 96.11 | **96.20** |
| | Three-frame densified | **92.52** | **99.52** | **98.63** | **98.60** | 78.49 | 80.10 | 80.98 | **99.22** | **98.17** | **97.98** |
| Part-$A^2$ | Raw Dataset | 87.09 | 99.41 | 90.48 | 90.35 | **81.28** | 80.54 | **80.86** | 98.14 | 90.18 | 90.06 |
| | One-frame densified | **88.85** | **99.48** | **96.26** | **96.51** | 80.87 | **80.81** | 79.94 | **98.74** | **95.86** | **90.11** |
| | Three-frame densified | **89.60** | **96.30** | **94.87** | **95.22** | 79.42 | 78.94 | 79.44 | **98.60** | **94.02** | **94.13** |

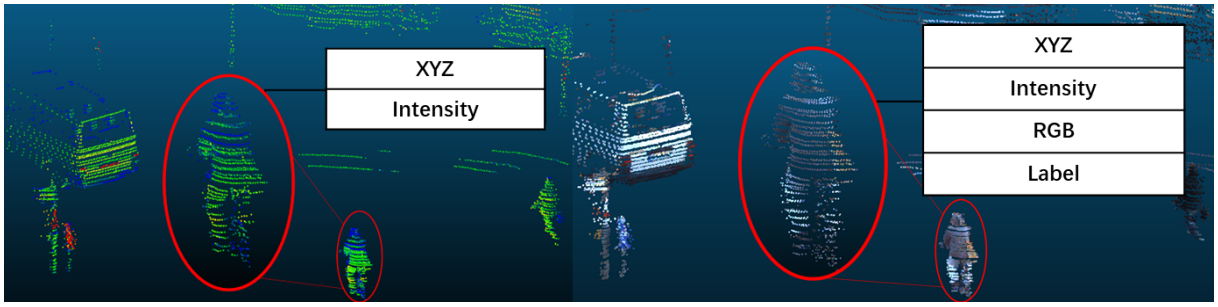


Fig. 4. Schematic diagram of dense semantic point cloud data structure.

TABLE IV
COMPARISON OF FIVE DIFFERENT LIDAR DETECTORS' PERFORMANCE ON DENSIFIED POINT CLOUD MULTI-CHANNEL DETECTION

| Method | Car | | | Pedestrain | | | Cyclist | | |
|---|---|---|---|---|---|---|---|---|---|
| | Easy | Mod. | Hard | Easy | Mod. | Hard | Easy | Mod. | Hard |
| PointPillars | 98.83 | 98.51 | 96.15 | 67.76 | 67.5 | 67.30 | 98.06 | 95.78 | 94.73 |
| SECOND | 98.21 | 97.42 | 96.15 | 76.15 | 74.28 | 74.69 | 94.78 | 92.11 | 92.09 |
| PointRCNN | 98.95 | 90.55 | 90.39 | 80.76 | 79.35 | 77.13 | 96.91 | 91.37 | 90.97 |
| PV-RCNN | 99.52 | 98.63 | 98.6 | 78.49 | 80.1 | 80.98 | 99.22 | 98.17 | 97.98 |
| Part-$A^2$ | 96.3 | 94.87 | 95.22 | 79.42 | 78.94 | 79.44 | 98.6 | 94.02 | 94.13 |

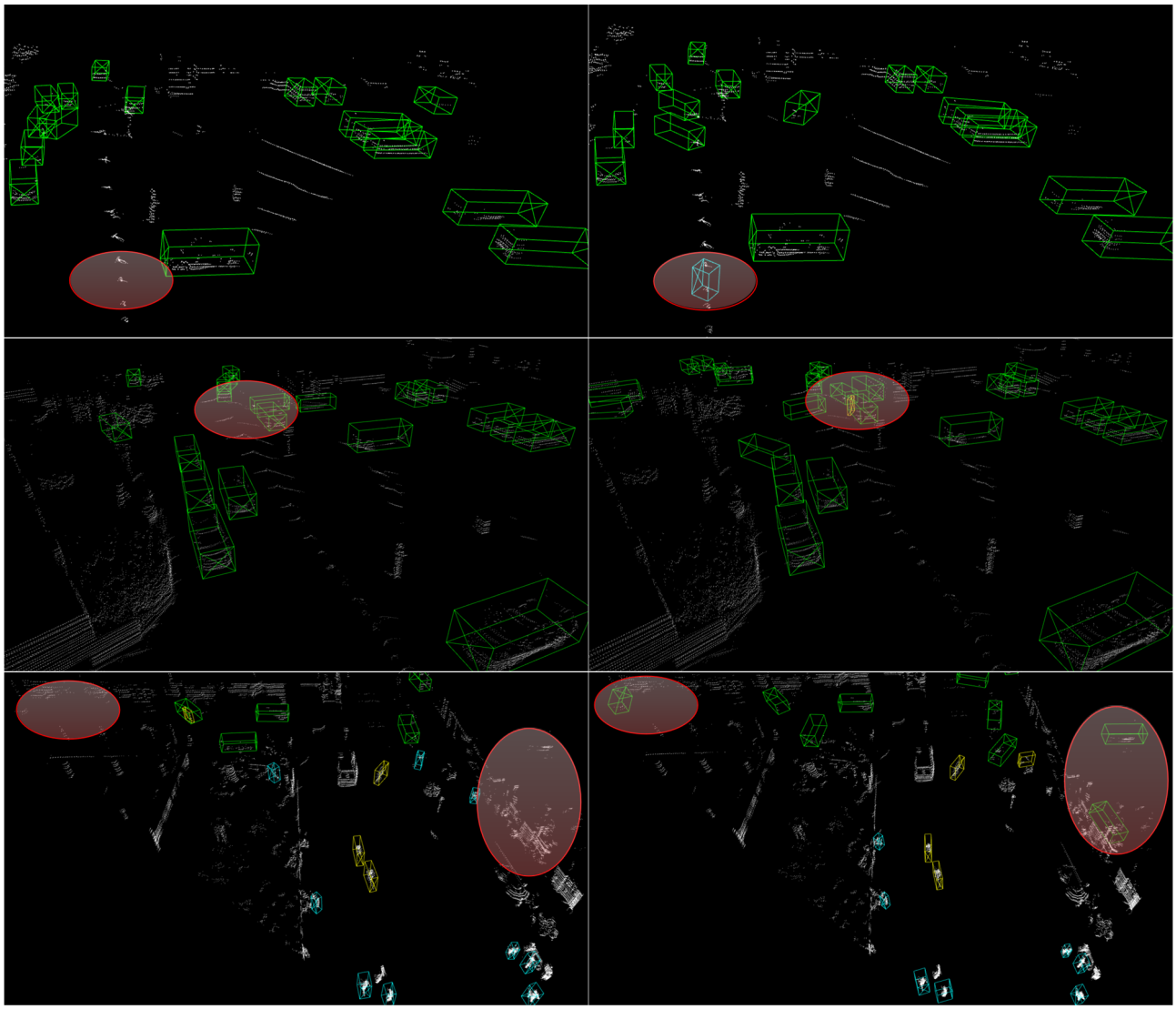

Fig. 5. Comparison of the 3D object detection results. The left column shows the results from the PV-RCNN detector trained on the three-frame densified dataset, and the right column shows the results from the PV-RCNN detector trained on the raw dataset. The boxes in the diagram are green for cars, light blue for pedestrians, and yellow for cyclists. Our proposed method effectively reduces the false positive detection in general scenes.

in dynamic scenes. Our method effectively overcomes the problem of localization errors in estimating long sequence scene flow and improves the accuracy and precision of shape completion. The localization accuracy of the scene flow estimation results can be effectively improved by introducing the Kalman filter to correct the position of the dynamic target. The densification results show that, compared with the ICP method, our method is more suitable for dynamic targets and achieves higher levels of accuracy and precision. Extended experiments on the KITTI 3D tracking dataset prove that our method effectively improves the LiDAR-only detectors' performance and achieves superior results to the baselines.

## REFERENCES

[1] C. Zheng, X. Yan, H. Zhang, B. Wang, S. Cheng, S. Cui, and Z. Li, "An effective motion-centric paradigm for 3d single object tracking in point clouds," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 43–60, 2024.

[2] M. Liu, "Robotic online path planning on point cloud," *IEEE Trans. Cybern.*, vol. 46, no. 5, pp. 1217 – 28, 2016.

[3] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Adv. neural inf. process. syst.*, vol. 30, 2017.

[4] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point cloud upsampling network," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2790–2799.

[5] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "EC-Net: An edge-aware point set consolidation network," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 386–402.

[6] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proc. Third Int. Conf. 3-D Digit. Imag. Model.* IEEE, 2001, pp. 145–152.

[7] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Robot. Sci. Syst.*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[8] K. Kim, C. Kim, C. Jang, M. Sunwoo, and K. Jo, "Deep learning-based dynamic object classification using lidar point cloud augmented by layer-based accumulation for intelligent vehicles," *Expert Syst. Appl.*, vol. 167, p. 113861, 2021.

[9] Z. Qin, H. Yu, C. Wang, Y. Guo, Y. Peng, S. Ilic, D. Hu, and K. Xu, "Geotransformer: Fast and robust point cloud registration with geometric transformer," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2023.

[10] H. Wang, Y. Liu, Q. Hu, B. Wang, J. Chen, Z. Dong, Y. Guo, W. Wang, and B. Yang, "Roreg: Pairwise point cloud registration with oriented descriptors and local rotations," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2023.

[11] W. Wang, M. R. U. Saputra, P. Zhao, P. Gusmao, B. Yang, C. Chen, A. Markham, and N. Trigoni, "Deeppco: End-to-end point cloud odometry through deep parallel neural network," in *Proc. IEEE Int. Conf. Intell. Rob. Syst.* IEEE, 2019, pp. 3248–3254.

[12] Z. Li and N. Wang, "Dmlo: Deep matching lidar odometry," in *Proc. IEEE Int. Conf. Intell. Rob. Syst.* IEEE, 2020, pp. 6010–6017.

[13] F. Huguet and F. Devernay, "A variational method for scene flow estimation from stereo sequences," in *Proc. IEEE Int. Conf. Comput. Vis.* IEEE, 2007, pp. 1–7.

[14] J. Quiroga, F. Devernay, and J. Crowley, "Local/global scene flow estimation," in *Proc. IEEE Int. Conf. Image Process.* IEEE, 2013, pp. 3850–3854.

[15] S. Hadfield and R. Bowden, "Kinecting the dots: Particle based scene flow from depth sensors," in *Proc. IEEE Int. Conf. Comput. Vis.* IEEE, 2011, pp. 2290–2295.

[16] J. Quiroga, F. Devernay, and J. Crowley, "Scene flow by tracking in intensity and depth data," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.* IEEE, 2012, pp. 50–57.

[17] X. Liu, C. R. Qi, and L. J. Guibas, "Flownet3d: Learning scene flow in 3d point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 529–537.

[18] X. Li, J. Kaesemodel Pontes, and S. Lucey, "Neural scene flow prior," *Adv. neural inf. process. syst.*, vol. 34, pp. 7838–7851, 2021.

[19] P. Li, R. Wang, Y. Wang, and W. Tao, "Evaluation of the icp algorithm in 3d point cloud registration," *IEEE Access*, vol. 8, pp. 68 030–68 048, 2020.

[20] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-icp: A globally optimal solution to 3d icp point-set registration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2241–2254, 2015.

[21] X. Shi, T. Liu, and X. Han, "Improved iterative closest point (icp) 3d point cloud registration algorithm based on point cloud filtering and adaptive fireworks for coarse registration," *Int. J. Remote Sens.*, vol. 41, no. 8, pp. 3197–3220, 2020.

[22] K. Koniarski and A. Myśliński, "A modified icp algorithm based on fast and optical flow for 3d registration," in *Proc. Conf. Comput. Sci. Intell. Syst.* IEEE, 2022, pp. 531–534.

[23] M. Vlaminck, H. Luong, and W. Philips, "Multi-resolution icp for the efficient registration of point clouds based on octrees," in *Proc. Fifteenth IAPR Int. Conf. Mach. Vis. Appl.* IEEE, 2017, pp. 334–337.

[24] M. Attia, Y. Slama, L. Peyrodie, H. Cao, and F. Haddad, "3d point cloud coarse registration based on convex hull refined by icp and ndt," in *Proc. Int. Conf. Mech. Mach. Vis. Pract.* IEEE, 2018, pp. 1–6.

[25] J. Yang, C. Wang, W. Luo, Y. Zhang, B. Chang, and M. Wu, "Research on point cloud registering method of tunneling roadway based on 3d ndt-icp algorithm," *Sensors*, vol. 21, no. 13, p. 4448, 2021.

[26] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[27] A. V. Phan, M. Le Nguyen, Y. L. H. Nguyen, and L. T. Bui, "Dgcnn: A convolutional neural network over large-scale labeled graphs," *Neural Netw.*, vol. 108, pp. 533–543, 2018.

[28] Z. Wang, S. Li, H. Howard-Jenkins, V. Prisacariu, and M. Chen, "Flownet3d++: Geometric losses for deep scene flow estimation," in *Proc. IEEE Conf. Comput. Vis.*, 2020, pp. 91–98.

[29] W. Wu, Z. Y. Wang, Z. Li, W. Liu, and L. Fuxin, "Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2020, pp. 88–107.

[30] Y. Wei, Z. Wang, Y. Rao, J. Lu, and J. Zhou, "Pv-raft: Point-voxel correlation fields for scene flow estimation of point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6954–6963.

[31] D. Simon, "Kalman filtering," *Embedded syst. program.*, vol. 14, no. 6, pp. 72–79, 2001.

[32] P. S. Maybeck, "The kalman filter: An introduction to concepts," in *Auton. Robot. Veh.* Springer, 1990, pp. 194–204.

[33] L. Winiwarter, K. Anders, D. Czerwonka-Schröder, and B. Höfle, "Full four-dimensional change analysis of topographic point cloud time series using kalman filtering," *Earth Surf. Dyn.*, vol. 11, no. 4, pp. 593–613, 2023.

[34] P. Słowak and P. Kaniewski, "Lidar-based slam implementation using kalman filter," in *Radioelectron. Syst. Conf.*, vol. 11442. SPIE, 2020, pp. 198–207.

[35] J. Xu, Z. Li, B. Du, M. Zhang, and J. Liu, "Reluplex made more practical: Leaky relu," in *IEEE Symp. Comput. Commun.* IEEE, 2020, pp. 1–7.

[36] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 605–613.

[37] G. Welch, G. Bishop *et al.*, "An introduction to the kalman filter," 1995.

[38] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *Int J. Rob. Resh.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[39] O. D. Team, "Openpcdet: An open-source toolbox for 3d object detection from point clouds," https://github.com/open-mmlab/OpenPCDet, 2020.

[40] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12 697–12 705.

[41] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[42] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 770–779.

[43] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10 529–10 538.

[44] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 8, pp. 2647–2664, 2020.