An Approach For Accurate Sceneflow Prediction For LiDAR-based Sensors.

Dhiraj D Shanbhag

MTech Project Report

Abstract

Simultaneous Localization and Mapping (SLAM) is considered a fundamental problem for robots to become truly autonomous. Having an accurate map enables us to do accurate localization and pose estimation. Generating a map by incrementally trying to register a point cloud over the map generated based on previous point clouds suffers from accumulated registration error. Dynamic objects in the scene are a major reason for such error accumulation. We want to use scene flow as a tool to mitigate this issue. Points belonging to these dynamic objects in the scene can be identified if accurate scene flow vectors are available. Our experimentation shows that the learning-based methods perform better on some parts of the scene, while ICP fares well in some other areas. Utilizing this observation, we propose a novel framework for accurate scene flow prediction. We segment the scene into multiple parts using the EMST based approach, then select the best method for each part. This method being unsupervised handles the problem of unavailability of annotated real-world training datasets. Our approach performs better than current approaches on the same KITTI dataset [1], even surpasses supervised setting performance for some recent approaches. We use combination of two approaches. We have shown that on KITTI, our approach performs better than these two approaches individually. The effectiveness of our approach is seen on real-world data collected at a warehouse $Refersection 6.7$.

1 INTRODUCTION

LiDAR-based SLAM performance is shown to provide reliable and realistic environment measurements, as it is less sensitive to lighting conditions. This enables it to extract robust features of the environment. Occlusion, dynamic objects in the scene are some factors that make SLAM challenging. The deep learning-based approach has outperformed classic algorithms for semantic computer vision tasks. However, the same advancement is not seen in the 3D geometric data processing problems, especially for registration and mapping problems [2, 3].

Deep learning-based approaches have shown promising results in point cloud classification and segmentation [4, 5]. Seminal papers by Qi et al. [4, 6] have proposed an approach to directly work on input point clouds, taking into consideration permutation invariance of points.

Scene flow is the displacement vector for each point in two consecutive frames. Accurate estimation of scene flow can aid in the detection of dynamic objects, motion planning, motion segmentation. Recent techniques learn to estimate the scene flow directly on 3D point clouds [7, 8, 9]. The majority of these methods require annotated datasets for training. Supervised methods are not suited for real-world scenarios as it is impossible to get a large annotated training set with ground truth scene flow. To compensate for this lack of real-world annotated data, learning-based methods use synthetic datasets for training like FlyingThing3D [10]. These networks are then tuned on real-world datasets. This has shown to improve prediction accuracy [1]. Some recent approaches have explored tuning on multiple datasets before evaluation of performance on dataset under consideration [1].

While these learning-based methods show performance improvement, traditional methods like ICP still performs better in some parts of the scene. Based on our observation on real-world datasets like roads and warehouses, ICP can predict the motion vectors more precisely for static objects.

We predict two sets of scene flow vectors each from the deep learning-based method and ICP. To combine these methods we break down the scene into instances using EMST based approach. The main task now is to identify the best among the two sets of scene flow vectors for each object. We use the chamfer distance-

based method for this selection.
The results show that our approach surpasses the deep learning-based approach and ICP by a huge margin. Detailed analysis is presented in Table 3 and Table 5. We also propose a method for combining these multiple approaches to get better performance than individual approaches. For the deep learning-based approach, we used the model trained on FlyingThing3D [9]. Our approach performs better than currently available self-supervised approaches on the KITTI dataset for scene flow prediction.

Our major Contribution:

- We propose a method to combine multiple methods of scene flow prediction to get better predictions.
- This EMST based approach for point cloud can be applied to other tasks. When multiple approaches with different properties are available. We have shown that such combinations have better performance than these individual approaches.
- We show the performance of the proposed method on real-world KITTI and warehouse LiDAR dataset. This shows the utility of the scene flow prediction.

We have explore techniques to use scene flow to identify dynamic objects in the scene for better mapping.

2 Related Work

Classical pose estimation techniques for LiDAR, based on methods like Iterative Closest Point (ICP) or variants of ICP [11] fail due to non-uniformity and sparsity of the LiDAR point clouds [2]. ICP is sensitive to initial poses. These iterative scan matching algorithms try to find the transformation that minimizes the distance between corresponding points in scans. The performance of these algorithms degrades in dynamic environments. Many techniques [5] that extract features from point clouds have been explored. These approaches use feature vectors for matching between scans and are less sensitive to scan quality but are computationally expensive. The 3D data is represented in different formats for processing, these include volumetric grids, range images, meshes, point clouds. Point clouds are currently gaining popularity as this preserves the geometric information in the scene without any discretization [12]. Unstructured nature of point clouds, high dimensionality poses challenges to the application of deep learning-based techniques on point clouds.
The approach by Ding et al. [3] considers only the temporal nature of the LiDAR scans while calculating the distance between the point clouds. A end-to-end learning-based 3D point cloud registration framework by Lu et al. [13] was able to achieve registration accuracy comparable to the state-of-the-art geometric methods. The learning-based keypoint detection and corresponding point generation algorithm [13] was able to achieve high accuracy in the learning-based registration task by using a combination of local similarity and global geometric constraints. The learning-based approach [2] has shown similar accuracy compared to that of state-of-the-art geometry-based approach for LiDAR odometry [14].
Many of the approaches convert the 3D point cloud data to 2D, then try to leverage the power of CNN. The [15] LiDAR scan is first converted to a 2D point map using the azimuth and elevation angles of points. This 2D data is then used as input to the CNN. These results help us infer that learning-based approaches can perform at an accuracy comparable to that of geometry-based approaches.
Qi et al. [4] proposed an approach to use a neural network that directly worked on input point clouds, taking into consideration permutation invariance of points. This network was able to do object classification, part segmentation, and semantic parsing. It learns a global point cloud signature by the aggregation of spatial encoding of each point. However, this didn't capture the structure induced by the metric. Qi et al. [6] improved on this by using a distance metric to partition the set of points into overlapping regions. This method extracted local features in small neighborhoods, these local features are further grouped to produce higher-level features similar to CNN. These papers proved to be the breakthrough in methods that work on point clouds directly. Deep learning-based approaches have shown promising results in point cloud classification and segmentation [4, 5] tasks.
Vedula et.al [16] introduced Scene flow as the three-dimensional motion field of points in the world. They proposed a framework for the computation of dense, non-rigid scene flow directly from optical flow. Estimating sceneflow using RGBD images generalize two-frame variational 2D flow algorithms to 3D [17]. Geometric cues available from the depth channel are used by some approaches [18, 19, 20]. Approaches for scene flow prediction on point clouds have also been explored. Dewan et al. [21] formulated this as an En-

ergy minimization problem based on matching SHOT feature descriptors for a subset of key-points. Ushani et al. [22] used iterative EM algorithm to estimate a locally rigid,non-deforming flow between successive occupancy grids. Recently, some deep learning-based techniques are deployed for scene flow prediction. Liu et al. [9] proposed an approach to estimate scene flow based on Qi et al. [6]. Flownet3D [9] estimates scene flow in a supervised manner from a pair of consecutive point clouds using an end-to-end learning-based method. It introduced a flow embedding layer that learns to correlate two point clouds and a set upconv layer that learns to propagate features from one set of points to the other. Flownet3D++ [8] improved over this by solving two problems present in Liu et al. [9]. First, the directions of predicted motion vectors differed significantly from ground truth vectors in their directions. Second, when deformable objects dominate the scene its performance degrades. This was achieved by adding the point-to-plane loss and cosine distance between predicted and ground truth scene flow vectors to the loss function. Mittal et al. [1] used a combination of cyclic consistency loss and nearest neighbor loss to introduce a self-supervised way to estimate scene flow. Wu et al. [23] estimates scene flow from two consecutive point clouds in a coarse-to-fine fashion. It introduces self-supervised losses that can train the network without any ground truth label. They proposed PointConv based cost volume layer that performs convolution on the cost volume without creating a dense 4-dimensional tensor. PointFlowNet [7] presented an end-to-end trainable model for joint 3D scene flow and rigid motion prediction and 3D object detection from unstructured LiDAR data.

3 Problem statement

Given two consecutive point clouds S and D.

$$S = \{s_i\}, \;\; i = 1, ..., N_s$$
$$D = \{d_j\}, \;\; j = 1, ..., N_d$$

where s_i and $d_j \in R^3$ are co-ordinates of points in point cloud. These point clouds don't necessarily have the same number of points. Also, correspondences between the points in point cloud might not exist. This is expected due to the presence of dynamic objects in the scene and viewpoint changes. Now, if a point $s_i \in S$ moves to a location $s_i^{'}$ in next frame. This $s_i^{'}$ need not be a point in the point cloud D. The translational motion vector is given by

$$m_i = s_i^{'} - s_i, \forall i = 1, ..., N_s$$

.
Our aim is to estimate motion vectors for each point in the point cloud S,

$$M = \{m_i\}, \;\; i = 1, ..., N_s$$

such that, distance between the point clouds $S^{'}$ and D is minimized. where,

$$S^{'} = \{s_i + m_i\}, \;\; i = 1, ..., N_s$$

4 Terminologies

4.1 Distance metric

We have considered two candidates for distance metric: Chamfer distance and EMD distance.

- Chamfer distance:
  If S_1 and $S_2 \subseteq R^3$, We define the distance between those two point clouds as,

$$d_{CD}(S_1, S_2) = 1/|S_1| * \sum_{x \in S_1} \min_{y \in S_2} \parallel x - y \parallel_2^2 +$$
$$1/|S_2| * \sum_{y \in S_2} \min_{x \in S_1} \parallel x - y \parallel_2^2$$

- Earth Movers distance:
  If S_1 *and* $S_2 \subseteq R^3$ *and* $|S_1| = |S_2|$
  We can define the EMD between S_1 and S_2 as,

$$d_{EMD}(S_1, S_2) = \min_{\phi : S_1 \to S_2} \sum_{x \in S_1} \parallel x - \phi(x) \parallel_2$$

  $\phi : S_1 \to S_2$ is a bijection.

In our approach, we use Chamfer distance for choosing best scene flow vectors among the available options.

4.2 Point cloud Segmentation:

Based on segmentation granularity we can briefly classify it into three categories:

- Semantic Segmentation
- Instance Segmentation
- Part Segmentation

In semantic segmentation, we try to identify subsets of points based on semantic meaning. Instance segmentation also requires identifying instances belonging to the same semantic class. It also identifies instance of a semantic class. Part segmentation involves segmenting 3D objects into their labeled semantic parts.

In our approach, we aim to divide the point cloud into subsets of points. Such that points belonging to each set form an object. We use a minimum spanning tree to find these sets. Experimental results on datasets show this approach is efficient.

5 Method

We use three major components in our pipeline:

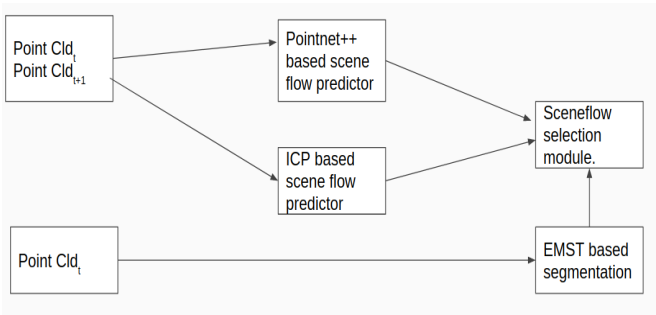


Figure 1: Our approach

5.1 ICP based component

ICP is used for aligning 3D point clouds. ICP finds the rotation matrix and translation vector for the source point cloud. This transformation reduces the distance between the point clouds. Let us consider consecutive point clouds S and D.

$$S = \{s_i\}, \quad i = 1, ..., N_s$$

$$D = \{d_j\}, \quad j = 1, ..., N_d$$

where, s_i and $d_j \in R^3$ are co-ordinates of points in point cloud.
These algorithms try to estimate $R \in SO(3)$ and $t \in R^3$, which will minimize the distance [24].

$$R, t = \underset{R \in SO(3), t \in R^3}{\operatorname{argmin}} \sum_{i=1}^{N_s} \parallel Rs_i + t - d_j \parallel_2$$

Then the scene flow is given by,

$$M = \{m_i\}, \quad i = 1, ..., N_s,$$

$$where, \ m_i = s_i - d_i \ and$$

$$d_i = \underset{d_k \in D}{\operatorname{argmin}} \parallel Rs_i + t - d_k \parallel$$

Given a relative rigid-body transformation, it iteratively refines it by finding correspondences between points in the two-point clouds. These algorithms are susceptible to local minima. Initial alignment plays a pivotal role. Rusinkiewicz et.al [25] compares variations of ICP, sampling strategies, and error metrics on different scenarios. These algorithms are considered as local registration methods as they have a dependency on an initial rough alignment. In our case, We need global registration algorithms that can work without initial alignment. These algorithms are known to produce less tight alignments. The alignments obtained from the global registration algorithm is used as initial alignment for local registration methods.

5.1.1 Global registration for initial alignment:

In consecutive lidar frames, we may not have the same number of points. To get a set of corresponding points, we use FPFHDBLP:conf/icra/RusuBB09 for global registration. The presence of outliers degrades the registration performance. This necessitates the use of techniques like RANSAC. We have experimented with FPFH [26] for global registration and RANSAC on its 33-dimensional feature space to get better alignment.

5.1.2 Fast Point Feature Histograms and RANSAC:

FPFH associates each point with a feature vector and these features are used to find correspondences. FPFH features are pose-invariant local features which depend on the underlying surface model properties for each point. 3D co-ordinates and normal at the point are considered for this feature vector. In the next step, we use these feature vectors to get the corresponding points. For each feature vector corresponding to a point in source point cloud, a 33-dimensional feature space of the target point cloud is queried for nearest neighbors. Some correspondences are rejected based on pruning techniques. Distance threshold between these corresponding points in 3-dimensional space constituting XYZ coordinates is used for pruning. Domain knowledge plays a major role in setting the distance threshold parameter. A valid set of corresponding points from the previous step is used to find the global alignment (6 DOF). This considerably improves transformation quality.

5.1.3 Refining the alignment:

The rigid transformation matrix obtained from FPFH and RANSAC can be refined. We experimented with ICP point-to-point and point-to-plane for registration The transformation obtained from FPFH along with

RANSAC is used as the initial transformation for this step. **TODO:**

Experiment	Fitment score
Random transformation matrix	
point-to-point	
point-to-plane	
FPFH	
FPFH + RANSAC	
FPFH + RANSAC + Local refinement	
point-to-point	
point-to-plane	
Fast global registration	
Fast global registration	
Local ref point-to-point	
Local ref point-to-plane	

Table 1: Fitment score comparision

5.2 Deep-learning based component

Deep learning-based approaches [9] take consecutive point clouds S and D as input, to predict the scene flow M. FlowNet3D [9] trains the network using a supervision signal, L1 loss (Huber loss) along with a cycle-consistency regularization is used.

This method requires a large dataset to train the model. Due to unavailability of large annotated real-world datasets with ground truth flow vectors, training is done on synthetic datasets [10] and tuned on real-world dataset. Flownet3D is based on Pointnet ++, it learns deep hierarchical features of point clouds. We use the model pretrained only on FlyingThings3D dataset using Flownet3D architecture.

5.3 EMST based segmentation:

The points belonging to an object in a point cloud are close to each other forming a cluster. Our aim is to get subsets of points belonging to individual objects.

- Construct the MST on the point cloud.
- Remove edges with edge length greater than a threshold.
- Points which still remain connected, can be considered as a same object.
- Noisy points can be removed by rejecting objects/cluster with less than a particular number of points.

Based on the scenario/dataset, we can tune these two parameters: Edge length threshold and Min-cluster size.

- Edge length threshold: Maximum permissible edge length in the minimum spanning tree.
- Min-cluster size: Minimum number of points that should be present in the subset/cluster to consider it as valid object.

5.4 Our Approach

Given consecutive point clouds S and D, we use the pretrained model, trained on FlyingThing3D [10] dataset by Flownet3D [9] approach without any fine tuning on kitti dataset. The sceneflow predicted by this model be,

$$SF_{learning} = \{sf\_learning_i\}, \;\; i = 1, ..., N_s$$

ICP is used to estimate $R \in SO(3)$ and $t \in R^3$, Which will minimize the distance between point clouds S and D. We calculate the scene flow by using ICP as follows,

$$SF_{icp} = \{sf\_icp_i\}, \;\; i = 1, ..., N_s$$

$$where, \;\; sf\_icp_i = R * s_i + t - s_i$$

We use EMST based approach to get disjoint subsets of points from S,

$$S = obj_{-1} \cup obj_1 \cup obj_2 \cup ..... \cup obj_k$$

where, obj_{-1} is subset of points which are not considered part of any valid object. Other objects given by obj_n, where $n = 1....k$, indicate set of points belonging to object n.
For each object obj, S_{obj} be points of S belonging to obj.
SF_{icp_obj} be sceneflow vectors of points belonging to obj predicted by ICP. $SF_{learning\_obj}$ be sceneflow vectors of points belonging to obj predicted by deep-learning based method. Now, based on predicted scene flow, we calculate the coordinates of the point in the next frame.
$Pred_{icp\_obj}$ indicates points of object in next frame as predicted by ICP. $Pred_{learning\_obj}$ indicates points of object in next frame as predicted by deep learning-based model.

$$Pred_{icp\_obj} = S_{obj} + SF_{icp\_obj}$$

$$Pred_{learning\_obj} = S_{obj} + SF_{learning\_obj}$$

Calculate distance between point clouds:

$$D_{icp} = Distance(Pred_{icp\_obj}, D)$$

$$D_{learning} = Distance(Pred_{learning\_obj}, D)$$

where, if $X = \{x_i\}, \quad i = 1, ..., N_x$ and $Y = \{y_j\}, \quad j = 1, ..., N_y$ then,

$$Distance(X, Y) = \sum_{i=1}^{N_x} \min_{y \in Y} \parallel x_i - y \parallel_2^2$$

$SF_{selected\_obj}$ indicates the final selected sceneflow for that object.

$$SF_{selected\_obj} = \begin{cases} \mathrm{SF}_{icp\_obj} & \mathrm{D}_{icp} \leq \mathrm{D}_{learning}. \\ \mathrm{SF}_{learning\_obj} & otherwise. \end{cases}$$

These steps are applied for every object to get scene flow vectors for every point of point cloud S.

6 Experiments

Dataset:

- FlyingThings3D [10] and KITTI [27].
- Carla Dataset.
- Dataset collected at a warehouse.

6.1 EMST based segmentation:

We can see in Figure:2, individual objects in the scene have been detected with satisfactory accuracy. The parameters for this approach are edge length threshold and minimum cluster size threshold. These parameters should be tuned based on domain knowledge. In scenarios like warehouse, objects will be in close proximity of each other compared to that of road dataset like KITTI. Edge length threshold for warehouse scenarios is lower compared to that on KITTI dataset. Figure 2(a) and 2(b), shows the performance of EMST on road dataset. Figure 2(c), shows the performance of EMST on warehouse dataset.

6.2 ICP vs learning based

ICP usually performs well in static areas like walls, while learning-based method fares well for dynamic objects. Experimental results have shown us that deep-learning approaches outperform geometric approaches in some tasks.

Parameters	ICP vs GT	Learning-based vs GT
ATE.rmse	2.52852 m	2.22314 m
ATE.mean	2.40179 m	2.08199 m
ATE.median	2.26976 m	2.06367 m
ATE.std	0.79048 m	0.77956 m
ATE.min	1.08893 m	0.64807 m
ATE.max	4.05035 m	3.91157 m

Table 2: ATE results on IISc Dataset.

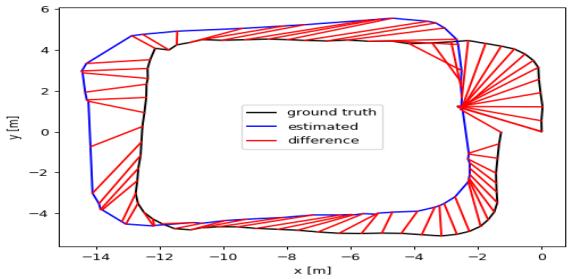


Figure 3: ATE: ICP

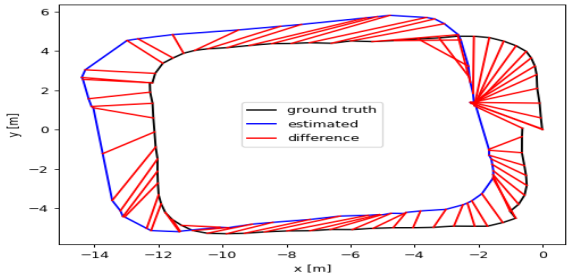
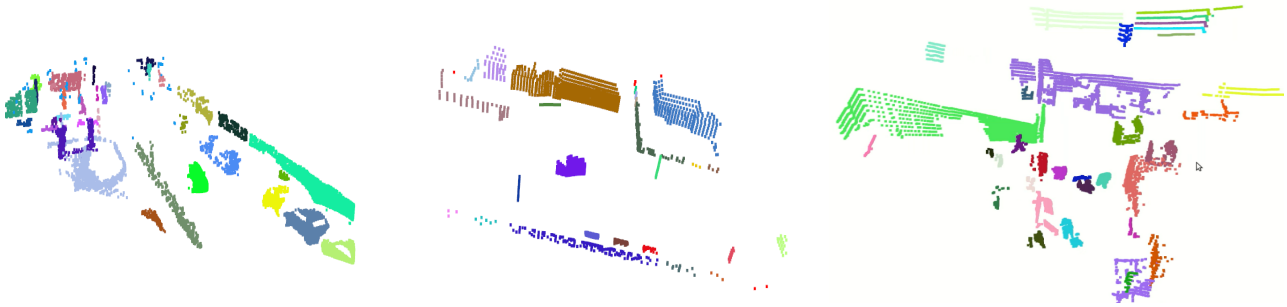


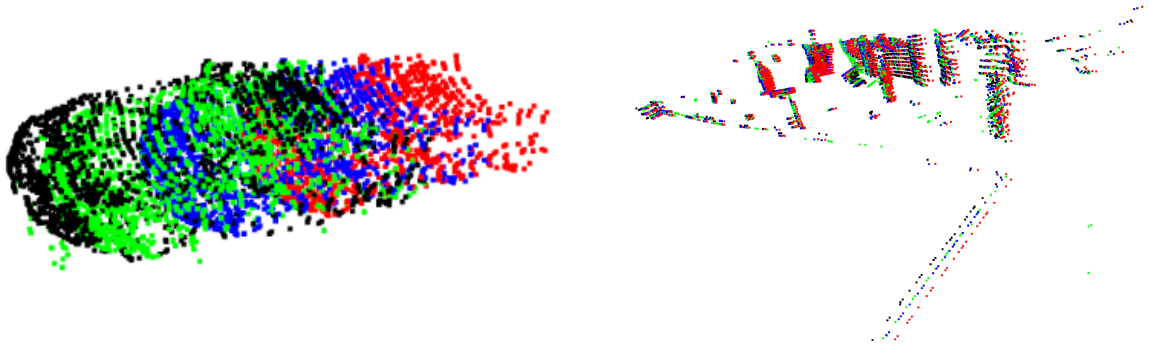
Figure 4: ATE: Deep learning-based

We have seen that ATE is less for learning based methods when compared to ICP in presence of dynamic objects in the scene. This behaviour is also seen in scene flow prediction performance. We have observed that deep learning-based technique usually over-estimates the scene flow for static objects in the scene. Figure 5(a), show the performance comparison on a moving car object in the scene. It shows a comparison between scene flow predicted by ICP, scene flow predicted by deep learning-based method, and the ground truth. We can observe that compared to blue points, black points have a better overlap on the ground truth green points for the car object. Figure 5(b), We can see that on the static wall surface, blue points corresponding to ICP based prediction better overlap over green points representing ground truth.



(a) KITTI dataset (b) CARLA dataset (c) Warehouse dataset

Figure 2: EMST based segmentation, Points belonging to an object in the scene have same colour. (random colouring used.)



(a) *Scan at time t* ,*Scan at time t+1* , *Scan at time t+1 predicted by ICP*, *Scan at time t+1 predicted by learning based*

(b) *Scan at time t* ,*Scan at time t+1* *Scan at time t+1 predicted by ICP*, *Scan at time t+1 predicted by learning based*

Figure 5: Comparison between learning-based and ICP. (a):Areas where learning based performs better, (b) Areas where ICP performs better.

6.3 Evaluation Metrics:

- EPE3D:
$$\| SF_\theta - SF_{GT} \|^2$$
averaged over each point.
SF_θ is the obtained scene flow.
SF_{GT} is the ground truth scene flow.
- Acc3DS: the percentage of points whose EPE3D < 0.05m or relative error $< 5\%$.
- Acc3DR: the percentage of points whose EPE3D < 0.1m or relative error $< 10\%$.
- Outliers3D: the percentage of points whose EPE3D > 0.3m or relative error $> 10\%$.

EPE3D is the main metric for scene flow performance evaluation.

6.4 Fast Point Feature Histograms and RANSAC

In Table 3, we can see that EPE3D has improved by 27% in the case of point-to-point ICP by using FPFH with RANSAC. This improved by 69% in case of point-to-plane ICP. We can also observe in row 6 to row 9, the performance of our approach improves when FPFH and RANSAC are used with ICP. Results improved by 58% in case of point-to-point and 63% in case of point-to-plane. The performance of FPFH varies based on the parameter used while selecting valid corresponding pairs between point clouds. All results shown in Table 4, follows our approach with combination of point-to-plane ICP and deep learning-based model.

Sl no	Experiment	EPE 3D ↓	ACC 3DS ↑	ACC 3DR ↑	outliers ↓
1	Deep-learning based model [9].	0.175043	0.09554347	0.405934	0.783591
2	point-to-point ICP without RANSAC	0.38246294	0.11862097	0.26319864	0.8280922
3	point-to-point ICP with FPFH+ RANSAC	0.27770174	0.680597	0.6852918	0.35532912
4	point-to-plane ICP without RANSAC	0.89796916	0.07424805	0.16215814	0.91517533
5	point-to-plane ICP with FPFH + RANSAC	0.27795105	0.67399797	0.68362812	0.36215297
6	Our approach:Deep-learning based + point-to-point ICP without RANSAC	0.14797779	0.19222829	0.54564061	0.66908076
7	Our approach:Deep-learning based + point-to-plane ICP without RANSAC	0.16729292	0.15317128	0.48157919	0.72421529
8	Our approach:Deep-learning based + point-to-point ICP with FPFH + RANSAC	0.06133774	0.70525882	0.83232568	0.27144086
9	Our approach:Deep-learning based + point-to-plane ICP with FPFH + RANSAC	0.06210524	0.69911642	0.83065674	0.27764179

Table 3: Effect of FPFH and RANSAC on performance

Sl no	Experiment	EPE 3D ↓	ACC 3DS ↑	ACC 3DR ↑	outliers ↓
1	0.04	0.10872504	0.47771841	0.67006789	0.45820222
2	0.08	0.06802063	0.67386144	0.81098575	0.30063782
3	0.12	0.06317567	0.69727	0.82841186	0.27458227
4	0.16	0.07174365	0.67549086	0.81514027	0.29352993

Table 4: Effect of distance threshold parameter used in FPFH + RANSAC.

6.5 Properties of an object:

Smoothness of a surface: Smoothness of a surface should preserved between consecutive scans for non-deformable objects. Such assumptions can be used to get slight performance gain. We calculate normal for each point in the point cloud using the points in its neighborhood. This can be used as a criteria while selecting sceneflow vectors for an object. For tuning the learning based network, sum of L2-norm of the difference between normal vectors of points and the mean can be used. We consider four neighbors to calculate the normal of a point. For each point p in the scan, let $\{n_1, n_2, n_3, n_4\}$ be its nearest neighbours.

$$normal\_vector = \sum_{j=1}^{3} \sum_{k=i+1}^{4} (n_j - p) \times (n_k - p)$$

$$normal(p) = normal\_vector / \parallel normal\_vector \parallel_2$$

smoothness loss for the set of points is given by:
P = $\{p_i\}$ where $i = 1.....n$ and $p_i \in \mathrm{R}^3$

$$smoothness(P) = \sum_{i=1}^{|P|} \parallel normal(p_i) - avgnormal(P) \parallel_2$$

where,

$$avgnormal(P) = \sum_{i=1}^{|P|} normal(p_i)/|P|$$

Given we have K objects, let P_k indicate set of points belonging to object k, and let SF_k indicate sceneflow vectors for those points.

$$P_k = \{p_{k,i}\} \; where \; i = 1.....|P_k|$$

$$SF_k = \{SF_{k,i}\} \; where \; i = 1.....|P_k|$$

where, $SF_{k,i}$ is a sceneflow vector corresponding to point $p_{k,i}$. Let

$$PredSF_k = \{SF_{k,i} + p_{k,i}\} \; where \; i = 1.....|P_k|$$

We expect $smoothness(P_k)$ and $smoothness(PredSF_k)$ should be same.
so,

$$s\_loss = \sum_{k=1}^{k=K} (smoothness(P_k) - smoothness(PredSF_k))^2$$

Properties of sceneflow vectors of an object: We try to exploit the properties of the objects for performance gain. Except in some cases, Every point that constitutes the object moves with a similar scene flow vector. We can use these properties to rectify some erroneous prediction of scene flow vectors.

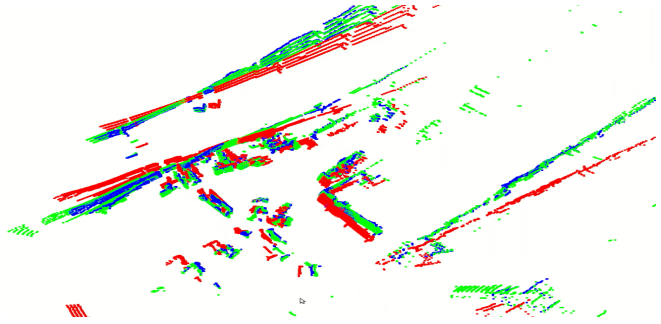


Figure 6: t^{th} *scan* ,$(t+5)^{th}$ *scan* , $(t+5)^{th}$ *scan predicted by our approach using scene flow.*

6.6 Results on KITTI dataset:

We compare the results shown by related papers on the KITTI dataset with our approach. We use the dataset provided by Flownet3D, which was created by converting the 2D optical flow to 3D scene flow. The ground points have been removed. The deep learning-based model is trained on FlyingThings3D dataset, with no fine-tuning done on KITTI dataset. KITTI dataset has 150 pairs of consecutive lidar frames with ground truth scene flow vectors. Number of points in point clouds vary, even two point clouds in a pair might have different number of points. The ground truth data has scene flow vector for every point in the first point cloud of every pair. Liu et al. [9] pointed out that downsampling introduces noise in prediction. So, multiple inferences needs to be done with random re-sampling of points then averaging the predicted flow vectors for each point. This strategy gives a slight performance gain.

6.7 Results on warehouse dataset:

This dataset is not public. The data collection was done using Ati Motors Mule device, mounted with Velodyne 16-beam LiDAR. In the warehouse scenario, objects are close to each other. The mule moves in comparatively lower speed in such environments compared to road scenarios. So, not much movement is seen between consecutive scenes. To solve this issue, we form pairs with t^{th} point cloud and $(t+5)^{th}$ point cloud. This increases the difficulty of the scene flow prediction problem but allows for a better visual evaluation. For better understanding the effectiveness of our approach, we will provide the videos of observation on this dataset. In Figure 6, we can see movement of walls, trolleys and people between the scene is captured by scene flow. Link for videos: **Videos**.

Sl no	Experiment	EPE 3D ↓	ACC 3DS ↑	ACC 3DR ↑	outliers ↓
1	Flownet3D [9] based model.	0.175043	0.09554347	0.405934	0.783591
2	Flownet3D ++ [8]	0.126	0.32	0.7364	
3	PointPWC-Net [23] (SELF SUPERVISED)	0.0694	**0.7281**	**0.8884**	0.2648
Just Go with the flow [1]					
4	KITTI (Supervised)	0.100	0.3142	0.6612	
5	KITTI (Self-Supervised)	0.126	0.3200	0.7364	
6	nuScenes (Self-Supervised) + KITTI (Self-Supervised)	0.105	0.4648	0.7942	
7	nuScenes (Self-Supervised) + KITTI (Supervised)	0.091	0.4792	0.7963	
Our Approach:					
8	(Pre-trained model+point-to-point ICP) (SELF SUPERVISED)	0.06133774	0.70525882	0.83232568	0.27144086
9	(Pre-trained model+point-to-plane ICP) (SELF SUPERVISED)	0.06210524	0.69911642	0.83065674	0.27764179
10	(Pre-trained model+point-to-plane ICP +point-to-plane ICP) (SELF SUPERVISED)	**0.06102659**	0.70486147	0.83212451	0.27238499

Table 5: Comparison of performance: Our approach vs other existing techniques.

7 Applications of sceneflow

7.1 Dynamic object detection

We can detect dynamic objects by utilizing the information about the movement of points between consecutive scenes. We can make use of the assumption that all points belonging to an object have a similar scene flow vector. Average of the norm of scene flow vectors might give some hints about the object's motion. Using this information dynamic objects can be removed from the lidar frames before mapping to improve mapping. This can be further used for object tracking, predicting objects location in upcoming frames, occlusion handling.

7.2 Mapping:

We used Deepmapping [3], to see the effect of dynamic objects on the mapping process. In the Figure 7, We observe clutter in the area between -15 to -7 on the horizontal axis in the first map. This is caused by the moving car in that region. After removing these dynamic object, the quality of map is improved.

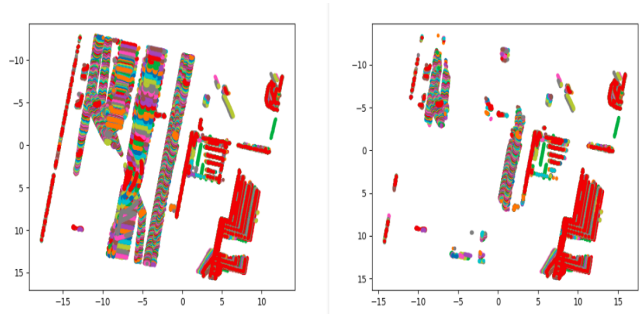


Figure 7: Effect of dynamic objects on mapping. Map generated on original scans vs after removing detected dynamic objects.

8 Conclusions

This approach to use techniques that are complementary in some parts of the scene. This approach can be utilized for performance improvement. As reported in Table 3 and Table 5, approaches exploiting these complementary behaviors can outperform these individual methods. We have shown these approaches produce the best result in comparison to currently available approaches. Downstream tasks like mapping, dynamic object detection, tracking can definitely build upon this approach. By using [3] framework we observed

the quality of the map can be improved by removal of dynamic objects.

9 Future Work:

Accuracy of tasks on LiDAR point clouds can be further improved by combining task specific DNN and geometric methods. As we pointed in our observation, we should explore combination of chamfer distance with other constraints during sceneflow selection for objects. Sceneflow as a tool can be utilized to track objects, next scene prediction. Improvement in the sceneflow prediction performance will increase accuracy in these downstream tasks. In our approach, we showed that specific techniques can be used to tackle specific areas. This is not constrained to DNN and ICP combination. Explore approaches to divide the scene to effectively use such combinations.

10 Acknowledgements:

I want to thank Prof.Chiranjib Bhattacharyya for his guidance throughout the project. I acknowledge support from Dr.Vinay for sharing his invaluable insights. I want to thank Mr.Sabyasachi Sahoo (RA, RBCCPS) for those discussions during the project. The mule requires a lot of expertise to operate, Iam grateful to engineers at ATI. They helped us collect data and shared their domain knowledge.

References

[1] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. *CoRR*, abs/1912.00497, 2019.

[2] Qing Li, Shaoyang Chen, Cheng Wang, Xin Li, Chenglu Wen, Ming Cheng, and Jonathan Li. Lo-net: Deep real-time lidar odometry. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8473–8482, 2019.

[3] Li Ding and Chen Feng. Deepmapping: Unsupervised map estimation from multiple point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8650–8659, 2019.

[4] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 77–85, 2017.

[5] Mingyang Jiang, Yiran Wu, and Cewu Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *CoRR*, abs/1807.00652, 2018.

[6] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5099–5108, 2017.

[7] 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, Long Beach, CA, USA, June 16-20, 2019*, pages 7962–7971. Computer Vision Foundation / IEEE, 2019.

[8] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Adrian Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*, pages 91–98. IEEE, 2020.

[9] Xingyu Liu, Charles R. Qi, and Leonidas J. Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 529–537. Computer Vision Foundation / IEEE, 2019.

[10] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4040–4048. IEEE Computer Society, 2016.

[11] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing ICP variants on real-world data sets - open-source library and experimental protocol. *Auton. Robots*, 34(3):133–148, 2013.

[12] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *CoRR*, abs/1912.12033, 2019.

[13] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. Deepicp: An end-to-end deep neural network for 3d point cloud registration. *CoRR*, abs/1905.04153, 2019.

[14] Ji Zhang and Sanjiv Singh. LOAM: lidar odometry and mapping in real-time. In *Robotics: Science and Systems X, University of California, Berkeley, USA, July 12-16, 2014*, 2014.

[15] Lucas Caccia, Herke van Hoof, Aaron C. Courville, and Joelle Pineau. Deep generative modeling of lidar data. *CoRR*, abs/1812.01180, 2018.

[16] Sundar Vedula, Simon Baker, Peter Rander, Robert T. Collins, and Takeo Kanade. Three-dimensional scene flow. In *Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece, September 20-25, 1999*, pages 722–729. IEEE Computer Society, 1999.

[17] Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D flow: Dense 3-d motion estimation using color and depth. In *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pages 2276–2282. IEEE, 2013.

[18] Xiuxiu Li, Yanjuan Liu, Haiyan Jin, Lei Cai, and Jiangbin Zheng. Layered RGBD scene flow estimation with global non-rigid local rigid assumption. In Jinchang Ren, Amir Hussain, Huimin Zhao, Kaizhu Huang, Jiangbin Zheng, Jun Cai, Rongjun Chen, and Yinyin Xiao, editors, *Advances in Brain Inspired Cognitive Systems - 10th International Conference, BICS 2019, Guangzhou, China, July 13-14, 2019, Proceedings*, volume 11691 of *Lecture Notes in Computer Science*, pages 224–232. Springer, 2019.

[19] Michael Hornacek, Andrew W. Fitzgibbon, and Carsten Rother. Sphereflow: 6 dof scene flow from RGB-D pairs. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 3526–3533. IEEE Computer Society, 2014.

[20] Julian Quiroga, Thomas Brox, Frédéric Devernay, and James L. Crowley. Dense semi-rigid scene flow estimation from RGBD images. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII*, volume 8695 of *Lecture Notes in Computer Science*, pages 567–582. Springer, 2014.

[21] Ayush Dewan, Tim Caselitz, Gian Diego Tipaldi, and Wolfram Burgard. Rigid scene flow for 3d lidar scans. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016*, pages 1765–1770. IEEE, 2016.

[22] Arash K. Ushani, Ryan W. Wolcott, Jeffrey M. Walls, and Ryan M. Eustice. A learning approach for real-time temporal scene flow estimation from LIDAR data. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pages 5666–5673. IEEE, 2017.

[23] Wenxuan Wu, Zhiyuan Wang, Zhuwen Li, Wei Liu, and Fuxin Li. Pointpwc-net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3d point clouds. *CoRR*, abs/1911.12408, 2019.

[24] Huu M. Le, Thanh-Toan Do, Tuan Hoang, and Ngai-Man Cheung. SDRSAC: semidefinite-based randomized approach for robust point cloud registration without correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 124–133, 2019.

[25] Anh Nguyen and Bac Le. 3d point cloud segmentation: A survey. In *IEEE 6th International Conference on Robotics, Automation and Mechatronics, RAM 2013, Manila, Philippines, November 12-15, 2013*, pages 225–230. IEEE, 2013.

[26] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, May 12-17, 2009*, pages 3212–3217. IEEE, 2009.

[27] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3061–3070. IEEE Computer Society, 2015.