DSLR : Dynamic to Static LiDAR Scan Reconstruction Using Adversarially Trained Autoencoder

Appendix

1 Implementation Details

1.1 LiDAR scan to Range image

The LiDAR scan obtained from VLP-64 LiDAR gives $\approx$ 1,30,000 points per scan. Geometrically, the scan consists of 64 laser beams having rotated 360°to give the point cloud its structure. We divide the azimuth angle into 512 bins while the elevation angle in 64 bins (due to 64 beams). This gives a grid structure of 64×512 where every cell consists of *x,y,z* coordinate calculated by averaging all points falling in the cell. It is to be noted that we can avoid the averaging by setting the number of bins for the azimuth angle equal to the number of points given by one laser beam in a 360°scan. We do not do this due to computational limitations. As observed in figure 1 from (Caccia et al. 2018), points far away from the origin for a scan are most noisy and are most difficult to train and thereby are removed. For more details refer (Caccia et al. 2018).

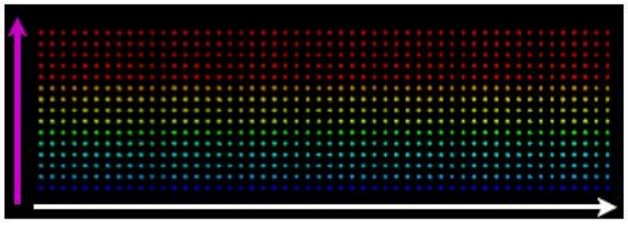


Figure 1: Image taken from (Caccia et al. 2018).Points at the same elevation angle have the same color. Every row denotes a 360°scan of the environment at a certain elevation angle. Every column denotes points at the same azimuth angle for different elevation angles.

1.2 Model architecture

Autoencoder We use an autoencoder inspired from (Caccia et al. 2018). The network uses a DCGAN based discriminator-generator as encoder-decoder. The input is LiDAR based range-image which is reconstructed at the output. The training procedure is shown in the Algorithm1.

Algorithm 1: AUTOENCODER

$E(\phi)$: DCGAN discriminator based Encoder
$G(\theta)$: DCGAN generator based Decoder
L : LiDAR frame based images
for $x \in L$ **do**
  $AELoss = 0$
  $x \xrightarrow{E(\phi)} r(x)$
  $r(x) \xrightarrow{G(\theta)} \overline{x}$
  $AELoss = MSE(x, \overline{x})$
  Train ϕ, θ using AELoss
end

Pair Discriminator Latent embeddings of a given LiDAR scan are obtained via an autoencoder. In a straightforward approach, one could use the above autoencoder and learn the proposed paired discriminator, that differentiates (static, static) vs (static, dynamic) LiDAR embedding pairs, by using the latent embedding given by it for static and dynamic frames. However, given that dynamic and static LiDAR scans look alike, except for the occlusions due to dynamic objects. The embedding space of both have a high overlap and simply using the above technique for training the paired discriminator doesn't perform well.

Let E_S and E_D be embedding space for static and dynamic LiDAR scans respectively. We hypothesize that $(E_S \cup E_D) - (E_S \cap E_D)$ differentiates static and dynamic scans. In other words, segmentation information can help differentiate the scans. However, as we do not use the segmentation information, we experimentally find that fixing the LiDAR latent space and training the discriminator doesn't yield the desired result. Letting the latent space change as the discriminator training progresses i.e., also training the autoencoder using reconstruction loss, helps the paired discriminator yield desired results. Therefore, without even using segmentation information, the discriminator can learn to focus on the regions of difference (occluded regions in dynamic frames) and is able to differentiate between (static, static) and (static, dynamic) LiDAR latent embedding pairs. This discriminator when used in adversarial training helps to map an embedding in dynamic latent-space to its corresponding static embedding in static latent-space. The train-

Algorithm 2: Discriminator

$DI(\gamma)$: Discriminator
S : Static LiDAR Images
D : Dynamic LiDAR Images
$G(\phi, \theta)$: Autoencoder for Static Frames
for $i \in range(0, |S|)$ **do**
  for $j \in range(0, |S|)$ **do**
    $DualLoss = LossDI = LossAE = 0$
    $S_i \xrightarrow{\phi} r(S_i) \xrightarrow{\theta} \overline{S_i}$
    $S_j \xrightarrow{\phi} r(S_j) \xrightarrow{\theta} \overline{S_j}$
    $D_j \xrightarrow{\phi} r(D_j) \xrightarrow{\theta} \overline{D_j}$
    $DI(r(S_i), r(S_j)) \rightarrow d_{SS} \in [0, 1]$
    $LossDI = BCE(D_{SS}, 1)$
    $LossAE = MSE(\overline{S_i}, S_i) + MSE(\overline{S_j}, S_j)$
    $DualLoss = \alpha * LossDI + LossAE$
    Backprop $DualLoss$ to train ϕ, θ, γ
    $DualLoss = LossDI = LossAE = 0$
    $DI(r(S_i), r(S_j)) \rightarrow d_{SD} \in [0, 1]$
    $LossDI = BCE(D_{SD}, 0)$
    $LossAE = MSE(\overline{D_j}, D_j)$
    $DualLoss = \alpha * LossDI + LossAE$
    Backprop $DualLoss$ to train ϕ, θ, γ
  end
end

ing procedure for the discriminator is shown in algorithm 2.

Adversarial Training We list the details of adversarial training in the algorithm. We use a value of α = 10 for the dual loss in the training procedure. The training procedure has been described in the Algorithm 3.

Inference **DSLR** inference is real time. **DSLR** takes 11ms for inference on a GTX 1080Ti while Velodyne LiDAR (used in all our datasets) take 100ms to generate a complete 360°scan.

1.3 Dataset Generation

Standard Point clouds e.g. ShapeNet(Chang et al. 2015) are different in structure and complexity from LiDAR point clouds. The number of points in a single VLP-64 LiDAR frame running at a frequency of 10Hz is ≈ 1,30,000, whereas a standard point cloud in ShapeNet dataset has ≈ 2600 points (Chang et al. 2015). LiDAR point clouds capture information at distances as far as 120m which is not the case with standard point clouds. Moreover, LiDAR point clouds have rich geometric and 3D information of a scene which is not the case with the arbitrary point clouds. Therefore we argue that using explicit static supervision for corresponding dynamic LiDAR frames is vital for a holistic reconstruction of a dynamic LiDAR scans to its corresponding static counterpart which may not be the case for standard arbitrary point clouds.
We detail an algorithm that extracts corresponding static and dynamic pairs of LiDAR scans using static and dynamic runs in the same environment using the same path.

Algorithm 3: Adversarial Training

$G_{static}(\phi_{SE}, \theta_{SD})$: Autoencoder taking static input
$G_{dynamic}(\phi_{DE}, \theta_{DD})$: Autoencoder taking dynamic input
$DI(\gamma)$: Discriminator
$\phi_{SE}, \theta_{SD}, \theta_{DD}, \gamma$: non-trainable weights
ϕ_{DE} : trainable weights
D : Set of Dynamic Frames
S : Set of corresponding Static Frames
for $i \in range(0, |S|)$ **do**
  for $j \in range(0, |S|)$ **do**
    $S_i \xrightarrow{\phi_{SE}} r(S_i) \xrightarrow{\theta_{SD}} \overline{S_i}$
    $D_j \xrightarrow{\phi_{DE}} r(D_j) \xrightarrow{\theta_{DD}} \overline{D_j}$
    $DI(S_i, D_{fj}) \rightarrow d_{SD} \in [0, 1]$
    $AdvLossDI = BCE(d_{SD}, 1)$
    $LossAE = MSE(\overline{D_j}, S_i)$
    $AdvDualLoss = \alpha * AdvLossDI + LossAE$
    Train ϕ_{DE} using $AdvDualLoss$
  end
end

We collect such data from CARLA Simulator as well as using our slow-moving Unmanned Ground Vehicle (UGV). The dataset generation method has also been explained in Algorithm4.

Dataset Generation with segmentation for DSLR++
Even after applying relative pose transformation in dataset

Algorithm 4: Paired Correspondence Dataset Generation

D : a random dynamic run is a set of dynamic LiDAR scans d_i in at pose p_i
S : a random static run is a set of static LiDAR scans s_j at pose q_j
$T_r(s_j)$: LiDAR scan s_j transformed rigidly based on pose r
$\triangle(p_i, q_j)$: relative pose difference between two LiDAR scans d_i and s_j
δ : desirable/given relative pose difference threshold for paired correspondence matching
M : training dataset for **DSLR** is a set of matching paired correspondence pair
Input: Run S and Run D
Output: M
$M \leftarrow \phi$
for $(d_i, p_i) \in D$ **do**
  for $(s_j, q_j) \in S$ **do**
    $r \leftarrow \triangle(p_i, q_j)$
    if $r < \delta$ **then**
      $M \leftarrow M + (d_i, T_r(s_j))$
    end
  end
end

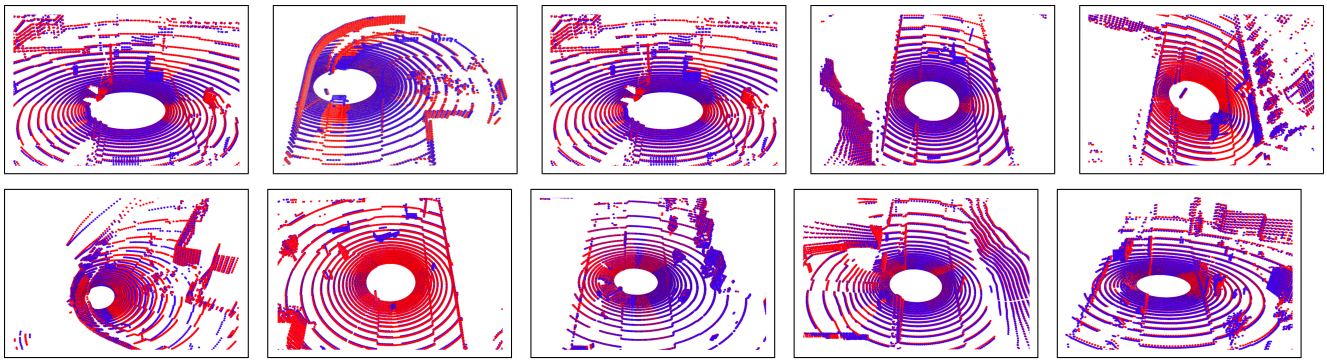
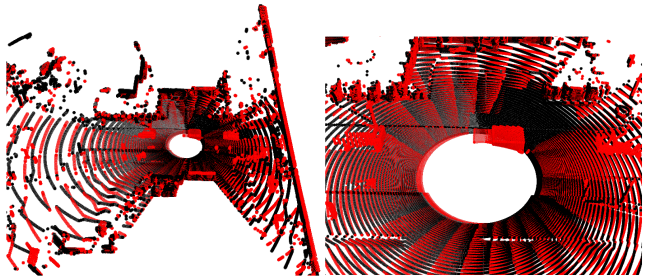
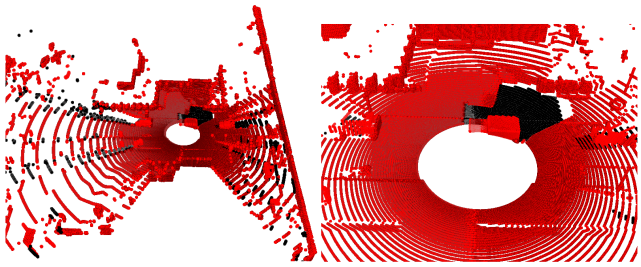


Figure 2: Examples of corresponding dynamic and static pairs from Carla-64 dataset. Best viewed in color. Dynamic and static scans are overlaid on each other to show the accuracy of correspondence. No segmentation information was used to create this dataset. Here, blue denotes dynamic scan and red denotes static scan. Regions with only red points indicate regions that are occluded in blue scans.



(a) Without Segmentation



(b) With Segmentation

Figure 3: Paired Correspondence Dataset created with and without segmentation. Best viewed in color. The right image is zoomed-in version of left image. Here, red denotes dynamic scan and black denotes corresponding static scan in the dataset. We can see that for dataset without segmentation, there is some mismatch between static points in both the scans. In the dataset using segmentation (which was used to train **DSLR++**), we can see no such mismatches and the static points in both the scans perfect overlay on each other.

generation algorithm, there can be some mismatch between static LiDAR scan points. This occurs because the dynamic and corresponding LiDAR scans were recorded at different places in the environment. Relative pose transformation can ensure that the static structures like walls overlap in both the scans but sometimes the exact LiDAR points don't overlap. Figure 3shows this difference between the dynamic and corresponding static frame.

This can lead to poor model performance since the model now is forced to learn on somewhat noisy static points. If segmentation information is available, we can overcome this issue using a technique similar to **DSLR−Seg**. Note that the corresponding static and dynamic frames have a degree of similarity equivalent to the percentage of static points in the dynamic frame. Table 4 shows that typically in real-world scenarios the amount of dynamism doesn't exceed 10%. This knowledge is leveraged to construct more accurate training data.

We use segmentation information to generate a mask to identify the points falling on dynamic objects. Using this mask, we only take regions corresponding to dynamic objects from corresponding static and regions corresponding to static points from the ground truth dynamic. Figure 3shows that corresponding static generated by this method has less error and all the static LiDAR points align perfectly in both the scans.

Ati Real-world Dataset (ARD)-16 generation We create first of its kind real world dynamic-static paired correspondence dataset collected using VLP-16 LiDAR. It was captured in outdoor environment at Robert Bosch centre, IISc with no moving objects during static run and several moving objects (1 car, 1 2-wheeler, few pedestrians) during dynamic run. It consists of 1.5k scans/run and we collected 10 dynamic and 5 static runs. This gave 14k LiDAR scan pairs for train, val and test. For busy urban areas, static runs can be obtained at night time.

1.4 LiDAR Reconstruction Evaluation Metrics

We evaluate the difference between model reconstructed static and ground truth static LiDAR scan on CARLA-64 and ARD-16 datasets using the following two metrics:

- **EMD**: Earth Mover's Distance is the minimum cost for transforming from one point cloud to the other and is given by:

$$dis_{\text{EMD}}\left(P_1, P_2\right) = \min_{\eta: P_1 \longrightarrow P_2} \sum_{x \in S_1} \|x - \eta(x)\|_2 \quad (1)$$

  Here, η represents 1-1 mapping between P_1 and P_2.

- **CD**: Chamfer Distance tries to capture the average mismatch between points in two given point clouds and is

Table 1: Comparison of the reconstruction baselines for various datasets. Lower is better.

Model	Uses Seg	Carla-64			ARD-16		KITTI-64
		EMD	Chamfer	LQI	EMD	Chamfer	LQI
AtlasNet	No	5681.85	5109.98	-	1464.61	176.46	-
ADMG	No	397.94	6.23	7.049	309.64	1.62	2.911
CHCP-VAE	No	343.98	9.58	4.080	88.94	0.67	1.128
CHCP-GAN	No	329.38	8.19	3.519	65.24	0.38	1.133
CHCP-AE	No	253.91	4.05	3.720	65.40	0.31	1.738
WCZC	Yes	$2.73 * 10^6$	478.12	-	-	-	-
EmptyCities	Yes	640.97	29.39	-	-	-	-
DSLR (Ours)	No	**232.51**	**1.00**	**3.350**	**57.75**	**0.20**	1.120
DSLR++ (Ours)	Yes	205.48	0.49	-	-	-	
DSLR-Seg (Ours)	Yes	**150.90**	**0.02**	-	-	-	
DSLR-UDA(Ours)	Yes	-	-	-	-	-	**1.119**

given by:

$$dis_{\text{CH}} = \sum_{x \in P_1} \min_{y \in P_2} \|x - y\|_2^2 + \sum_{y \in P_2} \min_{x \in P_1} \|x - y\|_2^2 \tag{2}$$

Here, P_1 and P_2 represents 2 sets of 3D points.

KITTI-64 does not have corresponding ground static, thus we propose a new metric LiDAR scan Quality Index LQI which estimates the quality of reconstructed static LiDAR scans by explicitly regressing the amount of noise in a given scan. This has been adapted from the CNN-IQA model (Kang et al. 2014) which aims to predict the quality of images as perceived by humans without access to a reference image.

1.5 Unsupervised Domain Adaptation

Let $x_s \in X_s$ represent data points from the source domain, $x_t \in X_t$ represent data points from the target domain, and $f(.)$ is the function used to map the data to a reproducing kernel Hilbert space (RKHS). The MMD is empirically approximated as follows:

$$MMD(X_s, X_t) = \left\| \frac{1}{|X_s|} \sum_{x_s \in X_s} f(x_s) - \frac{1}{|X_t|} \sum_{x_t \in X_t} f(x_t) \right\| \tag{3}$$

1.6 LiDAR scan Quality Index (LQI)

LQI has been inspired by the work done on No-Reference Image Quality Assessment where the visual quality of an image is evaluated without any reference image or knowledge about the type of distortion present in the image. We would like to adopt this task for LiDAR scans but evaluating LiDAR scans on their visual quality cannot a useful metric as it does not align with human visual perception nor it is the end goal for LiDAR reconstructions to be visually better. Therefore, in LQI, we evaluate reconstructed static LiDAR scans on the basis of the amount of noise injected by the model during the reconstruction. The assumption we use here is that this noise is Gaussian and hence, the LQI is a measure of the variance of the gaussian noise present in a normalized LiDAR scan. In this section, the use of the word noise would imply gaussian noise.

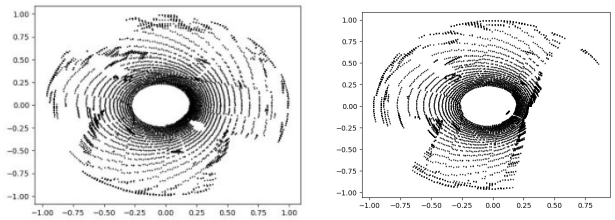
The model used for LQI is adopted from (Kang et al. 2014). The input LiDAR scan is normalized and passed through a convolutional layer with stride:-1 and a 7x5 kernel. This layer produces 50 feature maps, this is followed by a max pool and a min pool operation, after which each feature map is reduced to one max value and one min value. Two fully connected layers of 800 nodes after the pooling operation lead to a 1-dimensional output which is the LQI metric:- a measure of the variance of the noise present in the scan. The lower the LQI the lesser the noise. For training, Gaussian noise with varying parameters was added to LiDAR scans on the fly, and the model was expected to regress this variance and trained using the L1 loss between the predicted variance and the actual variance. Separate LQI models were trained using LiDAR scans from the KITTI-64 dataset and CARLA-64 dataset to score reconstructions on each dataset respectively.

The LQI can be considered as a good proxy to measure the quality of reconstruction for LiDAR scans for cases where ground truth is not available as we have already shown in the main paper that it correlates positively with Chamfer distance.

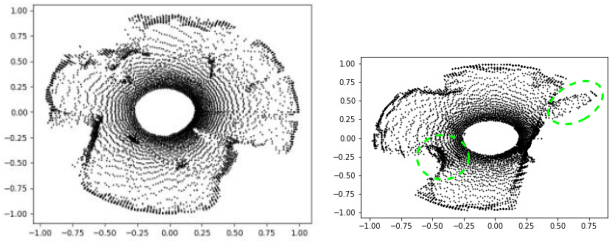
1.7 System Configuration and Training

For our experiments we use an Intel Core i9-9900K CPU @ 3.60GHz processor, loaded with NVIDIA GTX-2080Ti GPU. The autoencoder was trained for 600 epochs, the discriminator for 20 epochs while the adversarial framework was trained for 50 epochs. We use Adam Optimizer with a initial learning rate of $6e-4$, momentum of *0.1*, and weight decay of $1e-5$.

The LQI model was trained for 9 epochs. Adam optimizer was used with the initial learning rate of $1e-2$, momentum of *0.1* with learning rate reducing by 0.5 after every 3rd epoch.



(a) Raw dynamic scan



(b) Reconstructed Static Scan

Figure 4: Failure Case at Turns. The reconstruction obtained using **DSLR** is not able to reconstruct the details in the dynamic scan. Edges are thick instead of sharp detail as indicated in the encircled regions, as well as the general clarity and crispness is missing.

2 More LiDAR Reconstruction Results

2.1 Failure cases

We observe that reconstruction performs well when lateral movement of the vehicle is limited. Moderately occluded scenes are also reconstructed clearly in such cases. However, for cases like turns, road intersections model does not give a very accurate reconstruction. We also observe that it's hard for such a reconstruction algorithm to be generalizable. Testing with scans of a totally new region degrades the static reconstruction. This is because LiDAR scans in one region or area might be completely different from other areas and given the model has seen LiDAR scans of only a particular region or area, it might give sub-optimal results on input of a totally different region. Fig. 4 depicts the reconstruction at a turn. It is noticeable that LiDAR scan at turns are very different from normal straight motion scans and we observe that picking detailed structures is an issue in such cases.

3 SLAM Results and Analysis

Autonomous navigation systems which heavily rely on SLAM algorithms to localize and map in a known or unknown environment. Typically, SLAM algorithms are designed with the assumption that the environment is static. Violation of this assumption in real world like the presence of dynamic obstacles leads to a poor SLAM performance.

We confirm that dynamic environment results not only in high SLAM error as also seen in Table 2 but also creates maps of the environment with corruptions due to dynamic objects as shown in figure 6. The results for Table 2 were generated by running graph SLAM based LiDAR SLAM algorithm, Cartographer (Hess et al. 2016) on the dynamic and static runs in the same environment of our CARLA-64 dataset. Loop closure in graph SLAM methods typically helps in reducing SLAM error (like Absolute Trajectory Error (ATE)) in dynamic environments but we show in Table 2 that even with loop closure in Cartographer, SLAM error in dynamic environment is two fold compared to static environment. Manually tuning the parameters in Cartographer helps reduce the problem but doesn't solve it completely.

Therefore, instead of deleting these dynamic points, we propose to reconstruct the dynamic occlusion holes with the background static points which can significantly improve SLAM performance as per the figure 5. The metrics used for evaluation are described below

- **ATE** or Absolute Trajectory Error measures the similar-

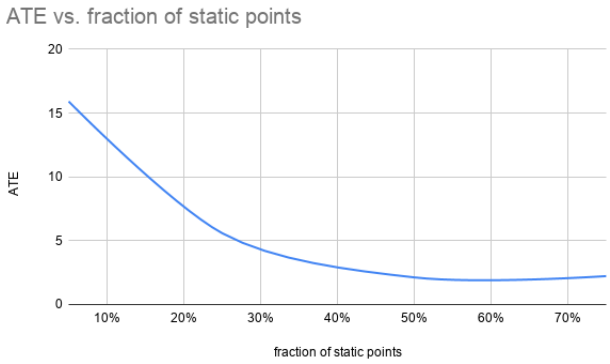


Figure 5: We experimentally study here the effect of reduced percentage of static points in a LiDAR scan. The fraction of static points were varied on X axis by randomly deleting points of individual LiDAR scans of a static run from CARLA-64 dataset. For a given fraction of static points, we evaluate SLAM error or Absolute Trajectory Error (ATE). It is seen that SLAM error doubles as the percentage of static points from a scene falls below 50%.

Table 2: SLAM error (ATE in in m) in static and dynamic environments

	w/o Loop closure	w/ Loop closure	w/ Loop closure & tuned
Dynamic	44.4	30.6	6.15
Static	13.3	11.4	4.07

Table 3: Detailed Comparison of DSLR-Seg with existing baselines on CARLA-64.

Carla Run	Model	ATE	drift	RPE Trans	RPE Rot
1	Pure-Dynamic	10.57	25.91	0.06	0.43°
	Detect & Delete (Model-Seg)	24.15	44.47	0.06	0.43°
	Detect& Delete (GT-Seg)	25.64	46.60	0.09	**0.42°**
	DSLR-Seg (Model-Seg) (Ours)	23.04	41.7	**0.057**	0.43°
	DSLR-Seg (GT-Seg) (Ours)	**8.53**	**22.08**	0.06	0.42°
2	Pure-Dynamic	4.72	4.25	0.05	0.41°
	Detect & Delete (Model-Seg)	2.49	2.61	0.062	0.50°
	Detect & Delete (GT-Seg)	2.49	**2.50**	0.058	0.41°
	DSLR-Seg (Model-Seg) (Ours)	2.14	2.8	0.05	0.38 °
	DSLR-Seg (GT-Seg) (Ours)	**2.07**	2.68	**0.05**	**0.24** °
3	Pure-Dynamic	15.79	25.58	0.06	0.37°
	Detect & Delete (Model-Seg)	**4.66**	9.53	0.06	0.37°
	Detect & Delete (GT-Seg)	6.84	21.20	0.05	0.37°
	DSLR-Seg (Model-Seg) (Ours)	7.12	**7.8**	0.05	0.37°
	DSLR-Seg (GT-Seg) (Ours)	11.39	16.15	**0.05**	**0.37°**

Table 4: Static dynamic points in each KITTI-64 run

Run	S%	SD%	MD%
0	91.07%	8.82%	0.09%
1	99.15%	0%	0.8%
2	97.6%	2.3%	0.07%
4	98.69%	0.36%	0.93%
5	96.51%	3.18%	0.29%
6	93.22%	6.68%	0.09%
7	89.2%	9.96%	0.08%
9	95.55%	4.21%	0.23%
10	98.03%	1.6%	0.36%

ity between shape of pose trajectory estimated by SLAM algorithm and is computes similar to (Sturm et al. 2012).

- **RPE** or Relative Pose Error also measures the pose error between two trajectories estimated by SLAM algorithm but it also is robust to accumulated global error. It is computed similar to (Sturm et al. 2012). *RPE* is computed differently for translational and rotational errors and is referred to here as **RPE trans** and **RPE rot**.
- **Drift** is the average of individual pose error w.r.t. ground truth pose and measures how far the robot is from its actual ground truth pose. Unattended high drift in robots can often lead to collision with surrounding objects.

We compare our proposed joint model approach for SLAM with existing baselines, which detect the dynamic points and delete it (Ruchti and Burgard 2018), (Vaquero et al. 2019), (Li et al. 2019),(Nagy and Benedek 2018) which we term as **Detect & Delete** in our results and with the base case with no LiDAR preprocessing namely **Dynamic**.

Our **`DSLR-Seg`** model and Detect & Delete baseline use dynamic segmentation to detect dynamic points, we report results with two variations of segmentations. They are reported with Model-Seg variation if U-net was used to give segmentation masks for dynamic points or GT-Seg variation is ground truth available mask was used segmentation. While Model-Seg variation shows the effectiveness of baseline and our method on a live vehicle, GT-Seg shows the effectiveness of Detect & Delete and our **`DSLR-Seg`** model in an ideal segmentation scenario where all the dynamic points of a given frame are perfectly known.

Table 3 reports the results of our method **`DSLR-Seg`** on Carla-64 dataset. The **`DSLR-Seg`** model was trained on this dataset and is finally evaluated on 3 long runs. We primarily test on runs containing loops because we would like our methods to use loop closure to correct for drifts developed over time. We wanted to show that SLAM algorithms fail in dynamic environments even with loop closure detection and our proposed method can give significant improvements over these methods. Our algorithm is improving the quality of reconstructed scan in dynamic setting which ensures a better chance of loop closure due to increment in matching score used for place recognition in loop closure detection. We can see that **`DSLR-Seg`** always outperforms the base case Dynamic method and is mostly better than the baseline method in most of the metrics, especially in ATE and drift.

Table 5: Detailed Comparison of **`DSLR-Seg`** with existing Baselines on KITTI-64.

Run	Model	ATE	drift	RPE trans	RPE rot
0	Pure-Dynamic	**7.137**	6.420	1.217	**1.772°**
	Detect & Delete (Model-Seg)	7.368	5.345	1.217	1.774°
	Detect & Delete (GT-Seg)	12.088	17.913	1.217	1.776°
	DSLR-Seg (Model-Seg) (Ours)	7.250	**4.530**	1.217	1.771°
	DSLR-Seg (GT-Seg) (Ours)	7.831	5.594	**1.217**	1.776°
1	Pure-Dynamic	40.229	68.429	3.187	1.155°
	Detect & Delete (Model-Seg)	39.627	**54.891**	3.188	1.155°
	Detect & Delete (GT-Seg)	52.589	82.683	3.179	1.158°
	DSLR-Seg (Model-Seg) (Ours)	146.648	169.260	**3.088**	1.157°
	DSLR-Seg (GT-Seg) (Ours)	**33.232**	60.215	3.204	**1.155°**
2	Pure-Dynamic	37.915	**28.623**	1.555	1.430°
	Detect & Delete (Model-Seg)	49.199	39.124	1.552	1.430°
	Detect & Delete (GT-Seg)	**12.483**	69.892	1.557	**1.429°**
	DSLR-Seg (Model-Seg) (Ours)	75.948	110.594	**1.547**	1.430°
	DSLR-Seg (GT-Seg) (Ours)	47.012	87.193	1.558	1.430°
4	Pure-Dynamic	0.565	**0.715**	2.055	0.192°
	Detect & Delete (Model-Seg)	0.268	1.078	2.050	0.192°
	Detect & Delete (GT-Seg)	0.588	1.026	2.049	0.192°
	DSLR-Seg (Model-Seg) (Ours)	0.598	1.208	2.054	0.192°
	DSLR-Seg (GT-Seg) (Ours)	**0.218**	0.785	**2.049**	**0.192°**
5	Pure-Dynamic	8.798	11.103	1.209	1.261°
	Detect & Delete (Model-Seg)	2.507	3.335	1.207	1.262°
	Detect & Delete (GT-Seg)	**1.815**	2.276	1.209	1.264°
	DSLR-Seg (Model-Seg) (Ours)	5.946	6.725	**1.204**	**1.261°**
	DSLR-Seg (GT-Seg) (Ours)	2.217	**2.211**	1.208	1.264°
6	Pure-Dynamic	1.783	2.576	1.632	1.514°
	Detect & Delete (Model-Seg)	2.941	4.891	1.631	1.514°
	Detect & Delete (GT-Seg)	**1.748**	**1.631**	**1.515**	1.776°
	DSLR-Seg (Model-Seg) (Ours)	2.424	4.281	1.632	1.514°
	DSLR-Seg (GT-Seg) (Ours)	3.687	4.148	1.630	**1.514°**
7	Pure-Dynamic	1.429	2.777	**1.025**	**1.646°**
	Detect & Delete (Model-Seg)	1.205	1.982	1.025	1.665°
	Detect & Delete (GT-Seg)	12.088	17.913	1.217	1.776°
	DSLR-Seg (Model-Seg) (Ours)	**1.075**	**1.555**	1.028	1.665°
	DSLR-Seg (GT-Seg) (Ours)	1.537	2.798	1.026	1.663°
9	Pure-Dynamic	**6.563**	8.628	1.551	1.352°
	Detect & Delete(Model-Seg)	7.732	7.544	1.551	1.353°
	Detect & Delete(GT-Seg)	7.995	9.436	**1.550**	**1.352°**
	DSLR-Seg (Model-Seg) (Ours)	7.136	**5.549**	1.551	1.352°
	DSLR-Seg (GT-Seg) (Ours)	7.647	7.656	1.552	1.353°
10	Pure-Dynamic	1.866	5.498	1.195	1.308°
	Detect & Delete (Model-Seg)	**1.629**	4.208	**1.192**	**1.303°**
	Detect & Delete (GT-Seg)	1.732	**2.938**	1.194	1.305°
	DSLR-Seg (Model-Seg) (Ours)	2.409	5.652	1.194	1.307°
	DSLR-Seg (GT-Seg) (Ours)	1.826	6.501	1.196	1.308°

We trained and evaluate our model **`DSLR`** on ARD-16 dataset and report our results in table 6 against base case Dynamic. Since ARD-16 dataset was collected by us, we

Table 6: Detailed Comparison of DSLR on ARD-16 dataset.

Run	Model	ATE	RPE Trans	RPE Rot
1	Pure-Dynamic	2.142	0.048	0.927°
	DSLR (Ours)	**2.117**	**0.045**	**0.904°**
2	Pure-Dynamic	6.939	0.051	0.823°
	DSLR (Ours)	**6.916**	**0.048**	**0.806°**
3	Pure-Dynamic	0.398	0.036	**0.457°**
	DSLR (Ours)	**0.395**	**0.035**	0.46°
4	Pure-Dynamic	0.230	0.027	**0.613°**
	DSLR (Ours)	**0.220**	**0.028**	0.612°
5	Pure-Dynamic	0.254	0.028	**0.442°**
	DSLR (Ours)	**0.241**	**0.027**	0.447°
6	Pure-Dynamic	0.245	0.028	**0.418°**
	DSLR (Ours)	**0.244**	**0.027**	0.460°

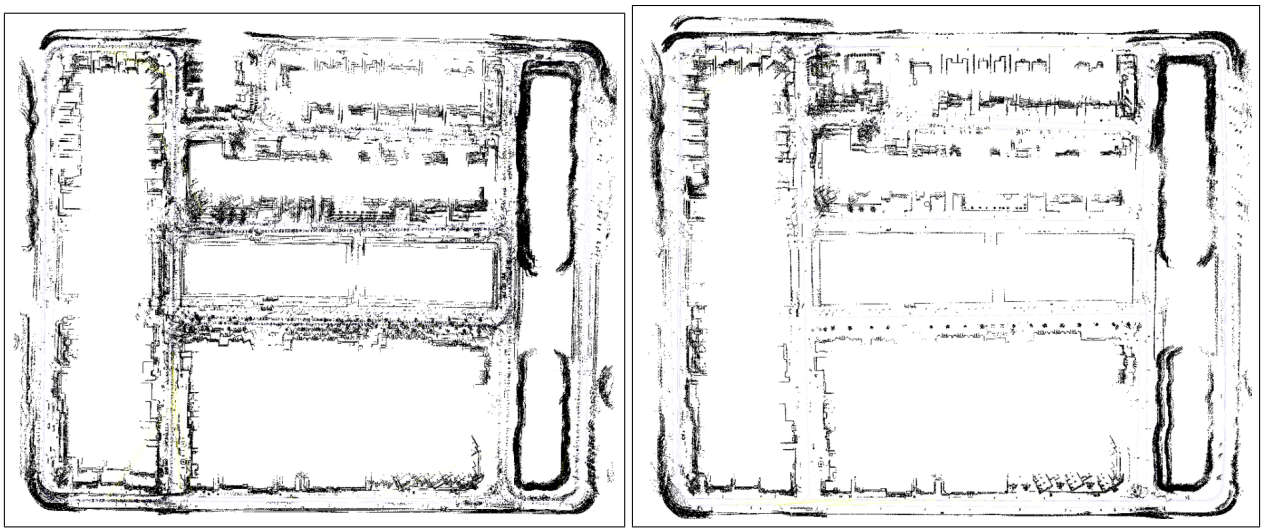


Figure 6: Map generated by SLAM algorithm (Cartographer) in the same CARLA environment for (1) Left: dynamic run (2) Right: static run. Although both the generated maps are not perfect, we can see that map generated from dynamic run has more corruptions compared to the map created from static run. Heavy corruptions in the map of an environment can result in degradation of localization performance over the time.

do not have ground truth segmentation mask. Therefore, we do not report SLAM performance for Detect & Delete and **DSLR-Seg** that require dynamic segmentation. In these results, we would like to bring out the fact that how in real world setting, segmentation masks may not be readily available. Despite this, we can not only collect data and train our proposed model **DSLR**, but it clearly outperforms the base case Dynamic in almost all the metrics especially ATE. However, we do note that the difference in result between **DSLR** and Dynamic is not quite high. This is because all the LiDAR frames in this run repeatedly observe the same part of the environment and Cartographer in both the cases is able to quickly correct its SLAM errors using loop closures. However, in a city like environment (similar to CARLA-64 or KITTI-64 datasets) such all-to-all loop closing constraints are not found often and our model will perform better than base case Dynamic.

In table 5, we compare our method against Dynamic and baseline method. A key thing to note is that our model was not trained on this dataset and we are reporting results using the **DSLR-Seg** model trained on CARLA-64 dataset. Despite this, we can see that our model **DSLR-Seg** is able to generalize to unseen dataset like KITTI-64 and our **DSLR-Seg** framework performs comparable or better to baseline methods in most of the cases.

We also report the percentage of dynamism in each these runs in Table 4. **S%**, **SD%**, **MD%** refers to average percentage of static points, average percentage of stationary dynamic points and average percentage of moving dynamic points in a given run, respectively. One surprising result we see that there are few runs like run no. 0, 9 where the base case Dynamic performs better than the baseline and our method. This was also reported by (Chen et al. 2019) and we can see that this happens when dynamic points are actually stationary during LiDAR frame capture. As a result, these dynamic runs get to see more number of static points compared to other methods and based on learnings from figure 5 this helps Dynamic to be the best performing method.

We report another surprising result that in these kind of runs, like run number 6, SLAM error is minimum for Model-Seg variation of **DSLR-Seg** compared to GT-Seg variation. We can understand this because in these runs, any error in dynamic segmentation can help the Model-Seg variation see more number of static points compared to its GT-Seg variation counterpart. But we would still like to detect all kind of dynamic points and replace them with static background because replacing any kind of movable points from the map helping in long term SLAM (Vaquero et al. 2019; Chen et al. 2019).

References

Caccia, L.; van Hoof, H.; Courville, A.; and Pineau, J. 2018. Deep generative modeling of lidar data. *arXiv preprint arXiv:1812.01180* .

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. 2015. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* .

Chen, X.; Milioto, A.; Palazzolo, E.; Giguère, P.; Behley, J.; and Stachniss, C. 2019. SuMa++: Efficient LiDAR-based semantic SLAM. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4530–4537. IEEE.

Hess, W.; Kohler, D.; Rapp, H.; and Andor, D. 2016. Real-time loop closure in 2D LIDAR SLAM. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 1271–1278. IEEE.

Kang, L.; Ye, P.; Li, Y.; and Doermann, D. 2014. Convolutional neural networks for no-reference image quality as-

sessment. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1733–1740.

Li, Q.; Chen, S.; Wang, C.; Li, X.; Wen, C.; Cheng, M.; and Li, J. 2019. LO-Net: Deep Real-time Lidar Odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8473–8482.

Nagy, B.; and Benedek, C. 2018. Real-time point cloud alignment for vehicle localization in a high resolution 3D map. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 0–0.

Ruchti, P.; and Burgard, W. 2018. Mapping with Dynamic-Object Probabilities Calculated from Single 3D Range Scans. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 6331–6336. IEEE.

Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; and Cremers, D. 2012. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 573–580.

Vaquero, V.; Fischer, K.; Moreno-Noguer, F.; Sanfeliu, A.; and Milz, S. 2019. Improving Map Re-localization with Deep 'Movable'Objects Segmentation on 3D LiDAR Point Clouds. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 942–949. IEEE.