Self Supervised Learning for Multiple Object Tracking in 3D Point Clouds

Aakash Kumar[1], Jyoti Kini[1], Ajmal Mian[2] and Mubarak Shah[1]

***Abstract*— Multiple object tracking in 3D point clouds has applications in mobile robots and autonomous driving. This is a challenging problem due to the sparse nature of the point clouds and the added difficulty of annotation in 3D for supervised learning. To overcome these challenges, we propose a neural network architecture that learns effective object features and their affinities in a self supervised fashion for multiple object tracking in 3D point clouds captured with LiDAR sensors. For self supervision, we use two approaches. First, we generate two augmented LiDAR frames from a single real frame by applying translation, rotation and cutout to the objects. Second, we synthesize a LiDAR frame using CAD models or primitive geometric shapes and then apply the above three augmentations to them. Hence, the ground truth object locations and associations are known in both frames for self supervision. This removes the need to annotate object associations in real data, and additionally the need for training data collection and annotation for object detection in synthetic data. To the best of our knowledge, this is the first self supervised multiple object tracking method for 3D data. Our model achieves state of the art results.**

I. INTRODUCTION

Multiple object tracking is an important task for autonomous systems to understand their surrounding environments and make appropriate decisions. It has many applications in autonomous driving and mobile robots, not only for navigation but also for meaningful and safe interaction with the environment. Multiple object tracking techniques have progressed significantly for the 2D video domain [1], [2], [3], [4]. The most common, and arguably the most successful, paradigm has been tracking by detection [1], [2], where objects of interest are first detected in individual frames and then correspondences are established between them to determine object trajectories across multiple frames.

However, multiple object tracking in the 3D domain, using point clouds, is still relatively new and faces multiple challenges. Firstly, point cloud data is unstructured and sparse, and has missing regions due to missing pulses in LiDAR and occlusions. Secondly, neural network architectures for object detection and affinity computation required for object correspondence in point cloud data are not as prevalent as they are for 2D videos. Finally, 3D point cloud datasets are very few and annotating objects in point clouds requires more effort compared to 2D videos, since it requires a sequence of translate, rotate and draw cube operations in 3D which is more time consuming.

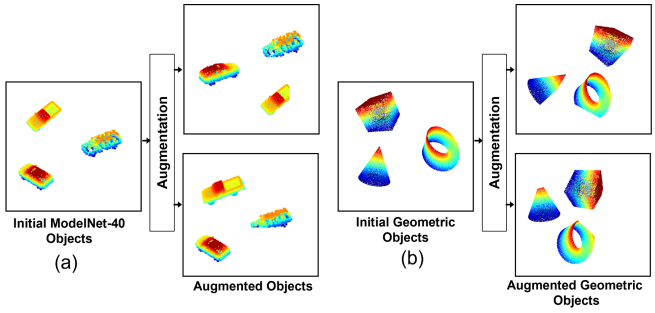


Fig. 1: Synthetic LiDAR frames generated from (a) CAD objects (ModelNet-40 data [5]), and (b) primitive shapes (cones, cubes etc). We place the objects at different locations in 3D space to create a synthetic LiDAR frame which is then augmented to form two frames with known ground truth movements of the objects. For augmentation, we use translation, rotation and object cut out (to simulate occlusion).

We propose a self supervised 3D multiple object tracking (SS3D-MOT) method to address the above challenges. Our method can effectively learn 3D object representations and their correspondences among frames in a self supervised manner from unlabelled real 3D point cloud data captured with LiDAR sensors as well as synthetic LiDAR frames generated from CAD (Computer Aided Design) models or even primitive geometric shapes as shown in Fig. 1. With self supervised learning, we can capitalize on the larger pool of unlabeled real LiDAR data. Moreover, the use of large amount of synthetic data ensures that our network does not overfit any particular dataset and can generalize across datasets.

Our approach follows the tracking-by-detection paradigm. The proposed SS3D-MOT network learns associations between two LiDAR frames in the form of an affinity matrix, which is followed by a min-cost-flow graph to compute the tracks. For real data, we use the JRDB [6] and KITTI [7] datasets. Given a LiDAR frame, we use the ground-truth detections to crop objects. Different augmentations (translation, cutout, rotation) are performed on the cropped objects to create two augmented frames which are used for self supervised training. Since both frames are created through augmentation, the ground truth object locations and associations are already known without the need for manual annotation.

*This work was conducted at UCF and supported by Lockheed Martin Corporate Engineering, Technology and Operations (CETO) University Engagement (UE) - Research. Professor Ajmal Mian is the recipient of an Australian Research Council Future Fellowship Award (project number FT210100268) funded by the Australian Government.

[1]Center for Research in Computer Vision, University of Central Florida, USA aakashjuseja@knights.ucf.edu, jyoti.kini@knights.ucf.edu, shah@crcv.ucf.edu

[2]University of Western Australia ajmal.mian@uwa.edu.au

We also create synthetic LiDAR frames to simulate multiple object tracking scenarios for self supervised training. For this, we use 3D CAD models from the ModelNet-40 [5] object classification dataset as well as primitive 3D geometric shapes such as cones, cubes and cylinders etc. We apply the same augmentation techniques and train our model. Note that for the real data, we require ground truth object detections, which can be obtained from an external off the shelf detector. However, using synthetic data (LiDAR frames generated from CAD models or primitive geometric shapes) we do not require any annotations for object detections or object associations as they are both known by design.

Our major contributions are summarized below:

- We propose a state-of-art SS3D-MOT model for multiple object tracking in 3D point clouds.
- We propose self supervised training techniques from real and synthetic data. In the former case, detected objects in a real LiDAR frame are augmented to generate two frames with known object associations. In the latter case, CAD models or primitive geometric shapes are used to generate two LiDAR frames with known object detections as well as object associations.
- We show that the proposed SS3D-MOT model is agnostic to the shape of the objects being tracked i.e. training on basic geometric shapes is as good as training on CAD models of the objects being tracked.
- We achieve state-of-art results.

II. Related Work

Multiple object tracking in 2D video can be categorized as online and offline methods. Recent online methods, mainly used for real time applications, compute the object-associations between current and previous frames using tracking-by-detection pipelines. After computing the affinity matrix, typically bipartite matching is applied on the affinity matrix to compute correspondences and generate tracks. These methods rely on learned discriminative features such as motion and appearance. Earlier works were based on hand-crafted features such as intersection over union of object bounding boxes [8] and appearance histograms [9]. Recent methods [1], [10], [11] involve deep-learning-based approaches for feature extraction to perform the tracking, whereas offline methods process the entire video sequence and apply global optimization [12], [9]. Recent works have focused on a unified framework for joint detection and tracking. One common approach [13], [14] to solve this problem is to build a MOT tracking architecture on top of a detector to predict the tracking offsets. Other methods use transformers to match tracklets [13].

Self Supervised Learning: Labeling video and LiDAR data is a laborious task and also prone to errors. More specifically, curation of training data [15], [16], [17] for supervised tracking involves immense human effort in bounding box annotations [18], [19], [20], [21], [22], [23] and the association of detections across frames. Expensive annotations and the lack of diversity in samples have been some of the major challenges in the area of supervised tracking. The basic idea of self supervised learning is to learn useful representations from large unlabelled data pools and fine tune with a much smaller labelled data. A popular solution is to define a pretext task with an objective function to learn useful data representations. Various pretext tasks are proposed such as image colorization [24], [25], [26], reordering patches [27], [28] and image in-painting [29].

Self Supervised learning for MOT in RGB videos: Owing to the expensive manual annotation requirements of supervised techniques, learning visual representations using predefined pretext tasks [30], [31], [32], [33], [34], [35], [36] in a self supervised manner has gained immense popularity in the area of tracking. However, most typical approaches [37], [38], [39] have not been able to fully simulate realistic appearance variations in tracking scenarios and, therefore, been ineffective in capitalizing on the utmost potential of self supervision in tracking. Recent approaches [40], [41] proposed for single object tracking exploit widely available unlabeled video data pools without fine-tuning. Other methods [42], [43] utilize the idea of transforming the color information between consecutive frames with the help of a matching network.

3D MOT: 3D object detection and tracking have made significant progress since the advent of seminal works in Point-Net architectures [44], [45], which have enabled efficient processing of unordered point cloud data. Some 3D MOT approaches employ off-the-shelf 3D detectors followed by Kalman filter based methods [46], [47] to track objects in 3D space. Other methods employ multi-modal data [48], [6] and extract 2D and 3D features to capture appearance and geometric features from registered images and point-clouds to perform 3D MOT. GNN3DMOT [49] employs motion information with LSTM neural network in parallel along with features from images and point clouds.

Despite the immense popularity of self supervised learning on images, there is limited work proposed for 3D point-cloud data. The most recent work [50] proposes a self supervised pre-training method to learn 3D features, where they show improved results on classification, detection and segmentation tasks. Whereas, we propose a novel approach, which is focused on self supervised tracking without a pretext task. Moreover, we show how synthetic frames can be used to train the proposed SS3D-MOT model in a self supervised setting for learning generic features, which makes the SS3D-MOT agnostic to the shape of the objects being tracked.

III. Methodology

The proposed self supervised 3D multiple object tracking (SS3D-MOT) model (see Fig. 2) adopts the tracking-by-detection paradigm and learns to predict the affinity matrix between two frames. Ground-truth detections are used to crop the objects (to be tracked) from a single LiDAR frame. The cropped objects are then augmented to generate a pair of augmented frames. Since the objects come from a single frame, the ground truth correspondences (affinity matrix) are known. SS3D-MOT uses object and box features together to generate the affinity matrix.

We use real LiDAR frames, one at a time, for self supervision. Hence, we do not require annotations for object associations between frames. However, the annotation of object detections with bounding boxes in individual frames is still required. Although, this can be done with an external off the shelf object detector, it makes the network learning sensitive to the errors of the used detector. Therefore, we also create synthetic LiDAR frames from CAD models or primitive geometric shapes, where the ground-truth detections as well as object associations are both known a priori. This not only eliminates the need for ground-truth associations, but also saves the cost of collecting real-world LiDAR data of moving objects and the need for an external detector for training.

Interestingly, our model learns generic 3D object features, and when trained on one object type, it still performs quite well on predicting the tracks of another object type. We show that SS3D-MOT trained on synthetic LiDAR frames generated from primitive geometric shapes performs well on the validation set of KITTI and JRDB real datasets to track cars and pedestrians.

A. Problem Formulation

In supervised tracking-by-detection, two LiDAR frames at time t and $t-1$ along with their ground-truth identity associations across the frames are required for training. We first eliminate the need for the ground-truth identity associations (affinity matrix) by training the proposed model in a self supervised setting. For this, we use a single LiDAR frame and generate two augmented frames with known identity associations.

Given a LiDAR frame along with ground-truth 3D object detections, which are publicly available with KITTI and JRDB datasets, we crop the objects and remove the background to get a frame A that consists of individual cropped objects, $\alpha_1, \alpha_2, \alpha_3, ...\alpha_n$. Frame A is of dimension $3 \times p \times N$, comprising the x, y, z coordinates of p points per object, and N is the number of objects to be tracked. We use random sampling to select $p = 256$ points for each object in our experiments. Next, we perform different augmentations on each object in frame A to create two augmented frames $B = \{\beta_1, \beta_2, \beta_3, ...\beta_n\}$ and $\Gamma = \{\gamma_1, \gamma_2, \gamma_3, ...\gamma_n\}$ with known identity associations. The augmented frames are used to train the model.

We perform three types of augmentations on 3D point cloud data: translation, rotation and cutout. *Translation:* To introduce displacement, we translate each object by 0.1, 0.2 or 0.4 meters randomly in the x, y and z directions. Any translated object n in frame B and Γ can be described as $\beta_{i=n} + [t_x, t_y, t_z]$ and $\gamma_{i=n} + [t_x, t_y, t_z]$, corresponding to the two augmented frames, (t_x, t_y, t_z represent translation along the x, y, z dimension). *Rotation:* Similarly, we incorporate angular motion by rotating individual objects. Each object is rotated randomly by 5°, 10° or 15° around the z-axis centered on the object. *Cutout:* To simulate occlusions, we cut a small portion of each object vertically by randomly selecting one of the vertical sides of the bounding box.

1) Real Data Augmentation for Self Supervised Learning: We transform a single frame from the real-world datasets, such as KITTI and JRDB, into two augmented frames, with known object associations. The object associations are then used to train our model.

2) Synthetic Data Generation for Self Supervised Learning: In real data, we eliminate the need for ground-truth identity associations through self supervision. However, we still require real data and ground-truth object detections obtained through manual annotation or with an external detector. To eliminate these two requirements as well, we propose two methods for generating synthetic LiDAR frames for self supervision using (i) synthetic CAD objects, or (ii) primitive geometric shapes.

Synthetic CAD Objects: We use CAD objects from the ModelNet-40 [5] object classification dataset to synthesize a LiDAR frame. A different object type such as car, chair, table, or monitor is randomly selected to synthesize a single LiDAR frame. After randomly selecting an object category, we randomly select 10 to 100 samples from that category and place them at different 3D locations to create a synthetic LiDAR frame. In Figure 1 (a), we show only a few objects for brevity. Since, we have placed the individual objects in the frame, we can easily determine their 3D bounding boxes using min-max in the x, y, z dimensions. Hence, we get a LiDAR frame A which is then converted into two augmented frames B and Γ to train our model.

Primitive Geometric Shapes: We also synthesize LiDAR frames containing primitive geometric shapes such as cones, cubes, cylinders, moebius, octahedrons, spheres, tetrahedrons, and torus. Figure 1 (b) shows a few example shapes used to generate a synthetic LiDAR frame. We randomly select a geometric shape, for example a cylinder, and generate 10 to 100 point clouds of that shape with random size parameters (height, width, radius etc) and random rotation, and place it at a random location in the frame. Then we repeat the process as mentioned above to generate two augmented frames.

B. Network Architecture

1) Feature Extractor: The proposed SS3D-MOT network architecture consists of object and box feature extractors and an affinity estimator. The object feature extractor takes two augmented frames B and Γ as input and extracts features of each object. The dimension of each input frame is $3 \times p \times N$, where p is the number of points per object and N is the number of objects. We set $p = 256$ in our experiments and put an upper limit of $N \leq N_m$, where $N_m = 100$ in our experiments. The feature extractor has five convolution layers, each with kernel size 1×1, with output channel dimensions of 64, 128, 256, 256 and 512. Hence, each input point in a given frame is converted to a 512-dimensional feature vector resulting in a $512 \times p \times N$ dimensional tensor. A max-pool operation is applied across the p dimension to get a single feature vector for each object.

The box feature extractor has 3 convolutional layers of kernel size 1×1 and channel outputs of 16, 32, and 64 re-

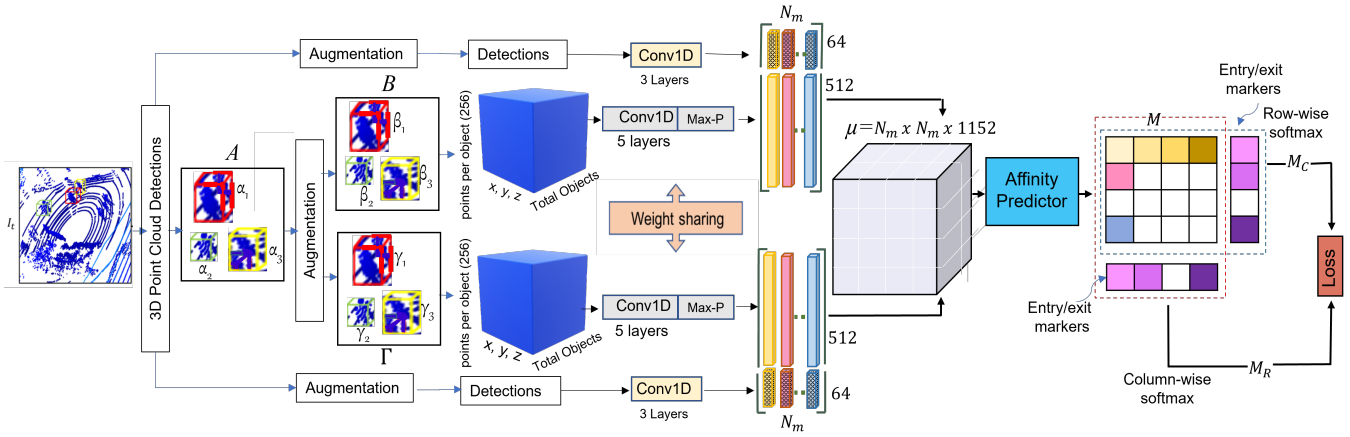


Fig. 2: SS3D-MOT Network Architecture: In self supervised learning, detected objects in a single LiDAR frame (real or synthetic) are cropped to create frame A without background. Objects in A are augmented with translation, rotation and cutout to generate frames B and Γ. Fixing the number of points per object to $p = 256$, a $3 \times p \times N$ matrix (where N is the number of objects) is formed to represent a LiDAR frame which is then passed through the object feature extractor that consists of five convolution layers followed by max-pooling along the p dimension. A box-feature extractor comprising 3 convolution layers extracts box features which are concatenated with object features. Combined features of N objects from two LiDAR frames are permuted to form an $N \times N \times (576 * 2)$ matrix which is passed through five convolution layers to gradually decrease the depth of μ and compute an $N \times N$ affinity matrix M. All convolution kernels in SS3D-MOT network are of size 1×1. To account for objects entering and leaving the scene, M is augmented with an extra column and row. Finally, row-wise and column-wise soft-max are performed to get M_C and M_R matrices used to compute the loss.

spectively. Along with the object features, the corresponding box features for every object are also used to learn the correct identity associations in the form of an affinity matrix. Box features provide explicit information about the center, length, width, height, and orientation of objects to the network for learning discriminative features. These box features are concatenated with the corresponding object features resulting in a combined feature vector of size 576 that represents a point-cloud object in 3D space.

2) Affinity Estimator: The affinity estimator converts the extracted features of two augmented frames into an affinity matrix M. To get this affinity matrix, feature vectors are arranged in a 3D matrix of size $N \times N \times (576 * 2)$ such that column vectors of the input matrices (object/box features from frame B and Γ) are concatenated in the depth dimension resulting in $N \times N$ permutations. When the number of objects N in the input frames is less than N_m (i.e. $N < N_m$), μ is padded to bring its size to $N_m \times N_m \times 1152$ which is then passed through the affinity predictor consisting of five convolutional layers of kernel size 1×1. The five convolutional layers gradually decrease the depth of the input matrix μ to compute the final affinity matrix M of size $N_m \times N_m$. The output feature maps of the affinity estimator are 1152, 512, 256, 128, 64, and 1. To account for objects entering and leaving the frames, we augment matrix M by adding an extra column and a row. Finally, row-wise and column-wise soft-max operations are performed to get M_C and M_R matrices which are then used for computing the loss.

C. Loss Functions

To train our model, we use two losses, forward-association loss (Eqn. 1) and backward-association loss (Eqn. 2). In Equations 1 and 2, G is the ground-truth of size $(N+1) \times (N+1)$. G_C and G_R are trimmed versions of G, where either the last row or column is ignored to compute the loss against M_C and M_R respectively. The forward association loss L_f is used to learn the correct identity association from frame B to Γ

$$L_f(G_C, M_C) = \frac{\sum(G_C \odot (-\log M_C))}{\sum G_C}, \tag{1}$$

and the backward-association loss L_b is used to learn the correct identity association from frame Γ to B

$$L_b(G_R, M_R) = \frac{\sum(G_R \odot (-\log M_R))}{\sum G_R}. \tag{2}$$

The network is trained using a combination of both losses

$$L = \frac{L_f + L_b}{2}. \tag{3}$$

IV. Experiments

A. Datasets

1) JackRabbot Dataset: JackRabbot dataset (JRDB) was collected by a custom-designed social mobile robot equipped with various sensors including, two LiDAR sensors (each with 16 channels), 360° cylindrical stereo RGB cameras, a 360° spherical RGB-D camera, a GPS and an IMU sensor to collect data in various indoor and outdoor scenarios. JRDB provides annotations for pedestrians which include 1.8 million 3D oriented bounding box annotations from

TABLE I: Results of our model on the validation sets of KITTI and JRDB datasets, when trained in supervised v/s self supervised setting.

Method	MOTA	ID Switches
JRDB Dataset		
Supervised	98.61	**2,165**
Self Supervised:		
☐ Real frame (JRDB Data)	**98.62**	2,209
☐ Synthetic frame (CAD objects)	90.45	15,891
☐ Synthetic frame (Primitive shapes)	90.90	13,060
KITTI Dataset		
Supervised	**79.45**	**31**
Self Supervised:		
☐ Real frame (KITTI Data)	78.58	55
☐ Synthetic frame (CAD objects)	73.16	43
☐ Synthetic frame (Primitive shapes)	72.56	87

two LiDAR sensors, and 2.4 million 2D bounding boxes in separate RGB images from different cameras. It also provides identity associations between the 2D and 3D bounding boxes and along the temporal dimension.

We only use the point cloud data from the two LiDARs and merge them, using calibration information, to get denser point clouds. There are a total of 27 training and 27 test sequences. We divide the training data into 20 sequences for train and 7 sequences for validation. To crop the objects, we use the provided 3D bounding box annotations, however, the model is trained for 3D tracking in a self supervised setting without using the temporal object associations.

2) KITTI Dataset: The KITTI dataset is captured by driving a car mounted with a Velodyne laser scanner, four cameras and a localization system. It contains outdoor data from a 39.2 kilometers drive, comprising around 200,000 3D object annotations (bounding boxes and ID associations) of cars and pedestrians. There are an average of 30 pedestrians and 15 cars per frame. This dataset has 21 training and 27 test sequences. We divide the training data to use 10 sequences for training and 11 for validation.

B. Implementation Details

We implement the proposed SS3D-MOT in the PyTorch library. We use batchsize of 1 given the large size of the point clouds. Since, smaller batch size increases the batch normalization error due to incorrect estimation of batch statistics, we use group normalization (GN) [51], which is independent of batchsize and remains stable for a wider range of batchsizes. GN computes the mean and variance within a group of channels to perform normalization. We utilize linear programming to find the optimal solution from min-cost flow graph. To solve linear programming, we use a mixed-integer-programming solver provided in Google OR-Tools.

C. Results on JRDB and KITTI Datasets

Although the proposed SS3D-MOT model can be trained in a self supervised setting, for comparison, we also train the model in a fully supervised setting using real data from the KITTI and JRDB datasets. For supervision, we use two consecutive frames along with their ground-truth detections (to crop the objects) and identity associations to train the model for multiple object tracking. We show MOT results on JRDB and KITTI datasets for different training methods in Table I. On both datasets, the results for self supervision using augmentation of real LiDAR frames are comparable to supervised results. Our model also achieves good results when trained on synthetic LiDAR frames generated from CAD objects or primitive geometric shapes. Note that we do not fine-tune the self supervised model on labelled data.

On the JRDB dataset, our model acheives 98.6 MOTA in supervised as well in self-supervised setting, and around 90 MOTA when our model is trained on synthetic data with CAD objects or primitive geometric shapes. We observe a similar pattern for the KITTI dataset. Figure 3 and 4 show visualizations of our results on the KITTI dataset for pedestrian tracking and car tracking respectively.

TABLE II: Ablation study of object features vs box features on JRDB. Box features perform better than object features and significant improvement is achieved when both are combined.

Object Features	Box Features	MOTA	ID Switches
✓	×	87.22	23,034
×	✓	90.26	2,878
✓	✓	**98.61**	**2,165**

TABLE III: Ablation study for self supervision: The effect of different augmentations when our model is trained on the real JRDB dataset or synthetic LiDAR frames.

	Translation	Cutout	Rotation	MOTA	ID switches
JRDB Dataset	✓	×	×	94.16	11,291
	✓	×	✓	96.55	4,312
	✓	✓	✓	**98.62**	**2,209**
Synthetic Data	✓	×	×	89.43	18,197
	✓	×	✓	90.29	16,442
	✓	✓	✓	**90.45**	**15,891**

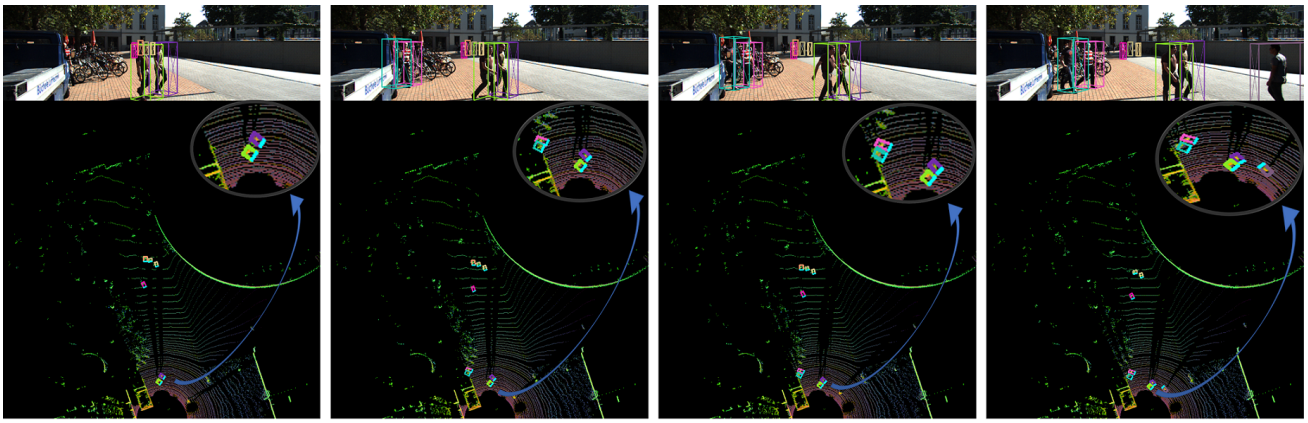
D. Ablation Studies

We use the JRDB dataset to perform several ablation studies of our proposed network architecture and the three approaches used for self supervision. We first show the ablation study for the two feature extractors in our network. Table II shows the results of object and box features individually. JRDB dataset contains sequences that are collected in densely packed scenarios such as the campus canteen. Therefore, using only object features results in higher identity switches whereas box features provide the explicit information of location, length, width, height and orientation information of each object. This drastically reduces the identity switches. Combining the box features and object features, we achieve higher MOTA of 98.61 and only 2,165 identity switches.

Table III shows how different augmentations contribute to

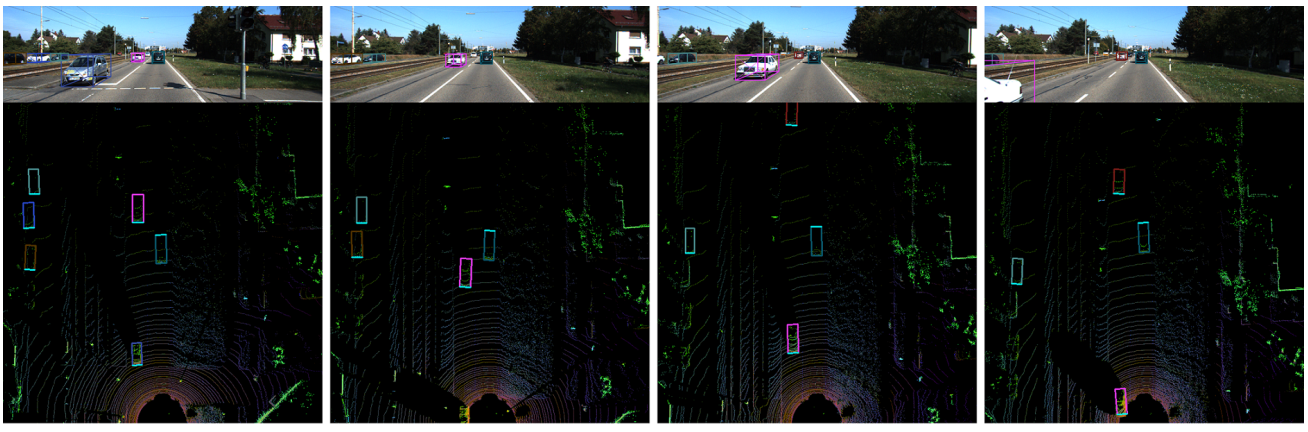
TABLE IV: Results from the JRDB Leaderboard. Our model achieves the best MOTA, IDF1 and HOTA scores on the leaderboard using self supervision with synthetic frames generated from CAD objects. Leader-board can be accessed using the following link. https://jrdb.erc.monash.edu/leaderboards/tracking

Method	MOTA	OSPA	IDF1	HOTA	Run Time (s)
PC-DAN [52]	22.56	0.99	13.89	15.56	0.12
JRMOT [6]	20.15	**0.98**	12.32	13.05	0.06
AB3DMOT [46]	19.35	**0.98**	10.67	11.81	**0.01**
SS3D-MOT (ours)					
Supervised	22.20	0.99	12.48	14.30	0.08
Self Supervised:					
☐ Real frame (JRDB Data)	19.80	0.99	7.91	9.67	0.08
☐ Synthetic frame (CAD objects)	**22.96**	0.99	**14.48**	**15.80**	0.08
☐ Synthetic frame (Primitive shapes)	17.83	0.99	6.64	8.33	0.08



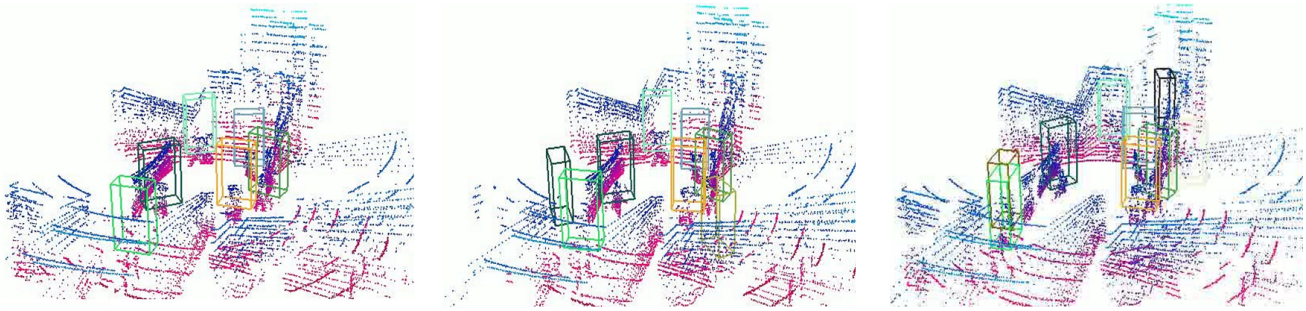
(a) Frame t (b) Frame t+4 (c) Frame t+8 (d) Frame t+12

Fig. 3: Qualitative results of our model (trained in self supervised manner) on KITTI pedestrian tracking using point clouds only. The RGB images on top, with ground truth bounding boxes, are shown only for illustration. Birds-eye view of LiDAR data is shown in the bottom row. All pedestrians are correctly tracked including the ones far away from the sensor (more obvious in LiDAR view).



(a) Frame t (b) Frame t+4 (c) Frame t+8 (d) Frame t+12

Fig. 4: Qualitative results of our model (trained in self supervised manner) on KITTI car tracking using point clouds only. The RGB images on top, with ground truth bounding boxes, are shown only for illustration. Birds-eye view of LiDAR data is shown in the bottom row where we can see that all cars are correctly tracked including the ones far away from the sensor (more obvious in LiDAR view). Cars move faster and cover large displacement across two frames compared to pedestrians. These results show that our method works well on fast moving objects.



(a) Frame t (b) Frame t+6 (c) Frame t+12

Fig. 5: Qualitative results of our model (trained in self supervised manner) on JRDB people tracking using point clouds only. Angular view of LiDAR data is shown where we can see that all people are tracked correctly (represented by colored boxes). Unlike KITTI, JRDB dataset provides 360 degree view covering more objects per frame.

the results in self supervised learning. In the case of real LiDAR frame augmentation, by applying translation only, we achieve 94.16 MOTA and 11,291 identity switches. By introducing rotations, the MOTA improves to 96.55 and the identity switches reduce from 11,291 to only 4,312. Introducing cutouts makes the model more discriminative, and further improves the MOTA to 98.62 and reduces the identity switches to 2,209. Hence, introducing rotation and cutouts improve MOTA by 4.46 and reduces identity switches by 9,082.

In the case of synthetic LiDAR frames made from CAD objects, we generate axis-aligned 3D bounding boxes of the objects by calculating their center, and min-max along each axis. This box is different from ground-truth bounding boxes of real data where the object orientation with respect to z-axis is also provided. For synthetic data, the bounding box may not tightly fit the object and the orientation is not available. Hence, we create pseudo orientations by rotating each object's bounding box. Because of these reasons, the improvements due to augmentations are not as large as in the case of the real data. Table III shows that, on synthetic data, translation alone achieves 89.43 MOTA score with 18,197 identity switches. Adding rotation and cutouts improves MOTA by only 1.02 and reduces identity switches by 2,306.

E. Results on the JRDB Leaderboard

We also submitted our results to the JRDB server for evaluation on the test data. Table IV shows that our proposed SS3D-MOT achieves the best MOTA, IDF1 [53] and HOTA [54] scores on the JRDB leaderboard when it is trained in a self supervised setting with synthetic LiDAR frames made from CAD objects. Figure 5 shows visualizations of our tracking results on the JRDB dataset. Leader-board can be accessed with the following link. https://jrdb.erc.monash.edu/leaderboards/tracking

V. Conclusion

We proposed a neural network model for multiple object tracking in 3D point clouds in a self supervised setting. We also proposed methods for self supervision, where we either augment a real LiDAR frame or generate synthetic frames. In the first case, we crop the detected objects and augment them to generate two frames with known object associations. In the second case, we use CAD models or primitive geometric shapes and augment these shapes to generate two synthetic LiDAR frames with known object detections and associations. We showed that our self supervised trained model achieves comparable results to the fully supervised one. Note that our self supervised model was not fine tuned on labelled data for a fair comparison between full supervision versus self supervision. Fine tuning our self supervised model on labelled data is likely to further improve its accuracy.

References

[1] S. Sun, N. Akhtar, H. Song, A. Mian, and M. Shah, "Deep affinity network for multiple object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 104–119, 2019.

[2] S. Schulter, P. Vernaza, W. Choi, and M. Chandraker, "Deep network flow for multi-object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6951–6960.

[3] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *European Conference on Computer Vision*. Springer, 2020, pp. 474–490.

[4] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," *International Journal of Computer Vision (to appear)*, August 2021.

[5] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[6] A. Shenoi, M. Patel, J. Gwak, P. Goebel, A. Sadeghian, H. Rezatofighi, R. Martín-Martín, and S. Savarese, "Jrmot: A real-time 3d multi-object tracker and a new large-scale dataset," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 10 335–10 342.

[7] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research*, 2013.

[8] E. Bochinski, V. Eiselein, and T. Sikora, "High-speed tracking-by-detection without using image information," in *14th IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2017, pp. 1–6.

[9] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[10] E. Baser, V. Balasubramanian, P. Bhattacharyya, and K. Czarnecki, "Fantrack: 3d multi-object tracking with feature association network," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1426–1433.

[11] J. Xu, Y. Cao, Z. Zhang, and H. Hu, "Spatial-temporal relation networks for multi-object tracking," in *IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3987–3997.

[12] S. Schulter, P. Vernaza, W. Choi, and M. Chandraker, "Deep network flow for multi-object tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2730–2739.

[13] Z. Lu, V. Rathod, R. Votel, and J. Huang, "Retinatrack: Online single stage joint detection and tracking," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 656–14 666.

[14] P. Bergmann, T. Meinhardt, and L. Leal-Taixé, "Tracking without bells and whistles," in *IEEE/CVF International Conference on Computer Vision*, 2019, pp. 941–951.

[15] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 621–11 631.

[16] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[17] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.

[20] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2008, pp. 1–8.

[21] F. Yang, W. Choi, and Y. Lin, "Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 2129–2137.

[22] P. Sun, R. Zhang, Y. Jiang, T. Kong, C. Xu, W. Zhan, M. Tomizuka, L. Li, Z. Yuan, C. Wang, *et al.*, "Sparse r-cnn: End-to-end object detection with learnable proposals," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 454–14 463.

[23] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6154–6162.

[24] M. M. Gustav Larsson and G. Shakhnarovich., "Learning representations for automatic colorization." in *European Conference on Computer Vision*, 2016.

[25] G. Larsson, M. Maire, and G. Shakhnarovich, "Colorization as a proxy task for visual understanding," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 840–849.

[26] R. Zhang, P. Isola, and A. A. Efros, "Split-brain autoencoders: Unsupervised learning by cross-channel prediction," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 645–654.

[27] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi, "Domain generalization by solving jigsaw puzzles," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2224–2233.

[28] M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash, "Boosting self-supervised learning via knowledge transfer," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9359–9367.

[29] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2536–2544.

[30] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1422–1430.

[31] N. Komodakis and S. Gidaris, "Unsupervised representation learning by predicting image rotations," in *International Conference on Learning Representations*, 2018.

[32] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European Conference on Computer Vision*. Springer, 2016, pp. 649–666.

[33] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2536–2544.

[34] I. Misra, C. L. Zitnick, and M. Hebert, "Shuffle and learn: unsupervised learning using temporal order verification," in *European Conference on Computer Vision*. Springer, 2016, pp. 527–544.

[35] B. Fernando, H. Bilen, E. Gavves, and S. Gould, "Self-supervised video representation learning with odd-one-out networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 3636–3645.

[36] R. Arandjelovic and A. Zisserman, "Objects that sound," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 435–451.

[37] A. Thabet, H. Alwassel, and B. Ghanem, "Self-supervised learning of local features in 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 938–939.

[38] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2088–2096.

[39] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik, "Learning shape abstractions by assembling volumetric primitives," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2635–2643.

[40] Z. Lai, E. Lu, and W. Xie, "Mast: A memory-augmented self-supervised tracker," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6478–6487.

[41] X. Li, S. Liu, S. De Mello, X. Wang, J. Kautz, and M.-H. Yang, "Joint-task self-supervised learning for temporal correspondence," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[42] A. F. S. G. C. Vondrick, A. Shrivastava and K. Murphy.

[43] S. Kong and C. Fowlkes., "Multigrid predictive filter flow for unsupervised learning on videos." *arXivpreprint arXiv:1904.01693*, 2019.

[44] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 77–85.

[45] H. S. Charles Ruizhongtai Qi, Li Yi and L. J. Guibas., "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108.

[46] X. Weng, J. Wang, D. Held, and K. Kitani, "3d multi-object tracking: A baseline and new evaluation metrics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 10 359–10 366.

[47] E. R. A. K. Samuel Scheidegger, Joachim Benjaminsson and K. Granström., "Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering." in *A IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 433–440.

[48] S. S. Z. W. J. S. Wenwei Zhang, Hui Zhou and C. C. Loy., "Robust multi-modality multi-object tracking." in *IEEE International Conference on Computer Vision*, 2019, pp. 2365–2374.

[49] Y. M. Xinshuo Weng, Yongxin Wang and K. Kitani., "Gnn3dmot: Graph neural network for 3d multi-object tracking with multi-feature learning." in *arXiv preprint arXiv:2006.07327*, 2020.

[50] Z. Zhang, R. Girdhar, A. Joulin, and I. Misra, "Self-supervised pretraining of 3d features on any point-cloud," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, October 2021, pp. 10 252–10 263.

[51] Y. Wu and K. He, "Group normalization," 2018.

[52] A. Kumar, J. Kini, M. Shah, and A. Mian, "Pc-dan: Point cloud based deep affinity network for 3d multi-object tracking (accepted as an extended abstract in jrdb-act workshop at cvpr21)," *arXiv preprint arXiv:2106.07552*, 2021.

[53] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *Computer Vision – ECCV 2016 Workshops*, G. Hua and H. Jégou, Eds. Cham: Springer International Publishing, 2016, pp. 17–35.

[54] Jonathon, A. Luiten, P. Osep, P. Dendorfer, A. Torr, L. Geiger, B. Leal-Taixe, and Leibe, "Hota: A higher order metric for evaluating multi-object tracking," in *International Journal of Computer Vision*, 2021, pp. 548–578.