Abstract

Multiple Object Tracking (MOT) is a challenging task and its application on Self Autonomous Driving Systems is crucial to ensure efficiency and accuracy in these systems. With the help of the Deep Affinity Network (DAN), we introduce the Point Cloud Deep Affinity Network (PC-DAN), which is a variant of the Deep Affinity Network (DAN) but on 3D point cloud LiDAR data. We conduct many experiments to judge the effectiveness of our point cloud model on the challenging KITTI benchmarks (i.e., Pedestrians and Cars) and report state-of-the-art performance on the Pedestrians benchmark.

1. Introduction

The idea of tracking multiple objects in a video is crucial for the success of Self Autonomous Driving Systems. In order for this to succeed we must ensure reliability and accuracy in our results [4]. In our Point Cloud Deep Affinity Network (PC-DAN) model, we propose a LiDAR only method, which utilizes 3D point cloud data provided by the KITTI 3D tracking dataset. Our model is a fusion of the multi-modality Multi-Object Tracking (mmMOT) framework [4], PointNet Architecture [2] and the Deep Affinity Network (DAN) [3]. The multi-modality Multi-Object Tracking (mmMOT) framework was used to help us load the LiDAR data from the KITTI dataset, PointNet helped us extract the features from the point cloud data, and the Deep Affinity Network (DAN) model was used to track the objects across two different frames separated by some interval $t$.

The Deep Affinity Network (DAN) harnesses the power of Deep Learning to represent multiple objects across different frames of a video and thus offers a solution for the challenging multiple object tracking problem. There are two components for the Deep Affinity Network (DAN): a Feature Extractor and an Affinity Estimator. Features are extracted from the images to learn certain representations of the specific objects being tracked. Affinity is computed across different frames to better associate the objects across multiple video frames, this is done to limit the number of ID-switches which is a major problem with multiple object tracking.

We have taken the Deep Affinity Network (DAN) [3] and applied it on the KITTI 3D tracking dataset. The Deep Affinity Network (DAN) model has shown state-of-the-art results on the challenging MOT17 dataset. Our Point Cloud Deep Affinity Network (PC-DAN) model has also demonstrated state-of-the-art results on the Pedestrians benchmark of the KITTI 3D tracking dataset. It is worthy to note that the MOT17 is a people tracking dataset, so the Deep Affinity Network (DAN) model produces better results on people rather than objects like cars. Although our Point Cloud Deep Affinity Network (PC-DAN) model accurately tracked pedestrians, it still falls a little bit short on the Cars benchmark of the KITTI 3D tracking dataset.

The rest of the article is organized as follows: in section 2 we will introduce the related works that aided in our model, in section 3 we will explain the Point Cloud Deep Affinity Network (PC-DAN) architecture, in section 4 we will explain how we trained on the Pedestrians and Cars benchmark, in section 5 we will reveal our results, and in section 6 we will conclude our novel work.

2. Related Work

Our work is mainly a cultivation of three major research papers that have defined the success of the Point Cloud Deep Affinity Network (PC-DAN) model. The first model we make use of is the multi-modality Multi-Object Tracking (mmMOT) framework [4], this is where we get the dataloader for the KITTI 3D tracking dataset from. The second major paper we reference is the PointNet model [2], which we used to extract features from the 3D point cloud LiDAR data. The third and final work we will reference is the Deep Affinity Network (DAN) model [3], which is essentially our entire model modified and pruned to fit the 3D point cloud LiDAR data.
3. Network Architecture

The Point Cloud Deep Affinity Network (PC-DAN) model is divided into three parts: Point Cloud-Based Feature Extractor, Exhaustive Feature Permutation, and Affinity Estimation.

3.1. Point Cloud-Based Feature Extractor

Unlike the Deep Affinity Network (DAN) \cite{3}, the input we pass into our model is composed of two parts, the two images of choice along with their labels, which is essentially a file with tracking info such as the class of the object we are tracking (i.e., Pedestrian, Car, Don’t Care...), the bounding box coordinates, etc. In order to accommodate for this, we decided to use the PointNet model \cite{2} as our feature extractor. In summary, we pass in two images as input each with their corresponding labels, and we just let PointNet do its work and extract the 3D features for us.

3.2. Exhaustive Feature Permutation

After we extract the features from the two input images and their corresponding labels, we end up with two separate matrices, 512x512 each, holding the features extracted from each image and its labels. As per the Deep Affinity Network (DAN) \cite{3}, the two matrices produced from the two inputs are convoluted and concatenated into one tensor. The concatenation of the two matrices helps us estimate the affinities between the two inputs and produce more accurate results for objects across different frames.

3.3. Affinity Estimation

The tensor resulting from the concatenation of the features extracted from the two inputs is passed into a compression network that is composed of 5 different convolutions of sizes 512, 256, 128, 64, and 1 respectively. This gradual dimension reduction along the depth of the tensor \cite{3} results in a matrix M that is now split into two separate matrices M\(_1\) and M\(_2\). We now apply a row-wise softmax on M\(_1\) along with a column trimming and a column-wise softmax on M\(_2\) along with a row trimming.

3.4. Loss Computation

The final step is to compute the loss. Loss is computed using four different loss functions. The first loss function is a forward direction loss that encourages correct identity association. The second loss function is a backward direction that ensures correct association. The third loss is a consistency loss rebuffing any inconsistency between forward and backward direction loss. The final loss is an assemble loss that suppresses non-maximum forward/backward association for affinity prediction \cite{3}.

4. Experiments

Several experiments were applied on our model to ensure optimal results.
4.1. Dataset

As mentioned, we used the KITTI dataset to test our model and obtained state-of-the-art results on it. The KITTI object tracking benchmark consists of 21 training sequences and 29 test sequences [1]. The 3D bounding box coordinates are labeled as tracklets in point cloud. In addition, KITTI object tracking benchmark also provides labels for the objects occlusion and truncation states [1]. For evaluating on the KITTI object tracking benchmark, only the Pedestrian and Cars benchmarks are available for this task, the Cyclist benchmark does not have enough instances for a comprehensive evaluation.

4.2. Training

Training on the Pedestrians benchmark was done as follows. We trained our model for a total of 200 epochs with a batch size of 1. The Machine Learning framework used to train this model is PyTorch and it uses one GPU. The images passed in are of size 224x224 pixels. Four different losses were used and we can refer to section 3.4 for more details on that.

4.2.1 Pedestrians

For the Pedestrians benchmark, there was not any hyperparameter finetuning applied and without a specified learning and dropout rate. Results on the Pedestrian benchmark top state-of-the-art results and this will be displayed in the section 5.

4.2.2 Cars

For the Cars benchmark, we went against our beliefs as for the Pedestrians and applied some hyperparameter finetuning to achieve our results. We used a dropout rate of 0.2 and a learning rate of $3 \times 10^{-4}$. The results of different dropout and learning rate experimentation will also be displayed in section 5.

5. Results

In this section we will display the results achieved on the Pedestrians and Cars benchmarks of the KITTI dataset [1].

the metrics we use to evaluate our results are the Multiple Object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP), Mostly Tracked, Mostly Lost, and ID-Switches.

We will display our results as follows, we will first show the training curves for each benchmark along with quantitative results for the metrics mentioned displayed in a table.

5.1. Pedestrians

The following graphs represent the training curves.
### 6. Conclusion

We have presented PC-DAN: Point Cloud Deep Affinity Network, which is a combination of the multi-modality Multi-Object Tracking (mmMOT) framework [4], PointNet Architecture [2] and the Deep Affinity Network (DAN) [3]. Our model was trained on the KITTI object tracking dataset and achieved state-of-the-art results on the Pedestrians benchmark. All of this was achieved with the unique combination of the Deep Affinity Network (DAN) [3] with PointNet [2], where PointNet extracts the 3D features and Deep Affinity Network (DAN) applies exhaustive feature permutation and computes the affinity between the two inputs. This is the first model to introduce 3D LiDAR data to the Deep Affinity Network (DAN) and it has proven its strength with its advanced results.

### References


---

**Table 2.** Quantitative results for Cars benchmark.

<table>
<thead>
<tr>
<th></th>
<th>MOTA</th>
<th>MOTP</th>
<th>Mostly Tracked</th>
<th>Mostly Lost</th>
<th>ID-Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cars</td>
<td>0.7919</td>
<td>0.8537</td>
<td>0.6667</td>
<td>0.0533</td>
<td>13</td>
</tr>
</tbody>
</table>

**Figure 5.** Cars Loss vs. Epochs graph