Attentional PointNet for 3D-Object Detection in Point Clouds

Anshul Paigwar[1,3] Ozgur Erkent[1,3] Christian Wolf[2,3] Christian Laugier[1,3]
[1]Univ. Grenoble Alpes, 38000, Grenoble, France
[2]Univ. Lyon, INSA-Lyon, CNRS, LIRIS, CITI-Lab, F-69621, Villeurbanne, France
[3] Inria-Chroma
{anshul.paigwar,ozgur.erkent,christian.laugier}@inria.fr, christian.wolf@insa-lyon.fr

Abstract

Accurate detection of objects in 3D point clouds is a central problem for autonomous navigation. Most existing methods use techniques of hand-crafted features representation or multi-sensor approaches prone to sensor failure. Approaches like PointNet that directly operate on sparse point data have shown good accuracy in the classification of single 3D objects. However, LiDAR sensors on Autonomous Vehicles generate a large scale point cloud. Real-time object detection in such a cluttered environment still remains a challenge. In this study, we propose Attentional PointNet, which is a novel end-to-end trainable deep architecture for object detection in point clouds. We extend the theory of visual attention mechanisms to 3D point clouds and introduce a new recurrent 3D Localization Network module. Rather than processing the whole point cloud, the network learns where to look (finding regions of interest), which significantly reduces the number of points to be processed and inference time. Evaluation on KITTI car detection benchmark shows that our Attentional PointNet achieves comparable results with the state-of-the-art *LiDAR-based 3D detection methods in detection and speed.*

1. Introduction

From high-speed Autonomous Vehicles that navigate on busy crossroads [1], to mobile robots that sweep the floor in your home [2], to humanoid robots that would serve you food in the restaurant, or quad-copters mapping and inspecting an industrial factory, and many other applications rely on three-dimensional (3D) data of physical surrounding. Accurately understanding the environment around them is crucial to their functioning in all these applications.

With the rapid development of Laser technology and availability of compact and affordable laser scanners (LiDARs), more 3D data is being captured and processed. In this work, we study one important task in 3D perception – 3D object detection, which classifies the object category and estimates the oriented 3D bounding boxes of physical objects in 3D space. Navigation of Autonomous Vehicles is one such principal application where high-resolution LiDARs are extensively used. LiDARs generate data in the form of point clouds representing the external surface, the geometry of the real-world objects. Unlike RGB images, point clouds are unstructured and how to interpret them to be used by deep learning architectures still remains an open problem. Recently Qi *et al.* in [3] proposed PointNet, a deep network architecture that can handle point cloud data directly without converting it into other forms of representation like images or volumetric grids. The simpler PointNet architecture, has shown impressive results on several tasks such as object classification and semantic segmentation.

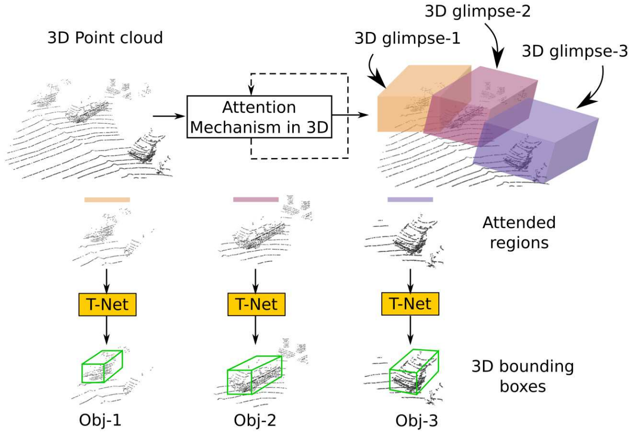

Figure 1. Attentional PointNet: 3D object detection in point clouds

Although PointNet can classify the whole point cloud, an adaptation to instance-level 3D object detection is not straightforward. Also, another limitation is that in the original work proposed by [3], the maximum number of points in the point cloud has been limited to 4096; however, a typical point cloud from a LiDAR contains more than 100k points. Therefore, it is an open challenge how to use PointNet with larger point sets.

Dealing with very large clouds is possible by cutting it into smaller regions processed separately. Some very recent multi-sensor approaches [4] [5] have proposed to first use the state of the art 2D detectors with RGB image and then projecting the detected 2D bounding boxes into 3D space to reduce the search space. Variants of PointNet are finally used for regressing the corresponding 3D bounding boxes. However, the need for an additional camera that is time synchronized and calibrated with the LiDAR restricts their use and makes the solution more sensitive to sensor failure modes.

Recently Simon *et al.* in Complex- YOLO [6] proposed an approach to project point clouds into birds-eye-view RGB height-map and used a modified YOLO 2D object detector to regress 3D bounding boxes. Converting point clouds into height-map does retain the height information but the structural information of the object is lost. For example, structures whose vertical projection is similar to cars can easily be miss-classified and result in false positives.

Currently, VoxelNet [7] and SECOND [8], are the two deep networks which directly use 3D LiDAR data without converting them into other 2D representations and output 3D bounding box predictions for multiple objects in uncontrolled environments. SECOND outperforms the state-of-the-art LiDAR-based 3D detection methods by a large margin. However, these architectures are complex and require a high amount of computation in order to run in real-time.

This work focuses on exploring alternative methods for detection based on LiDAR data only. We aim to design an efficient but simple architecture providing real-time performance on lower compute capability hardware. To this end, we propose to use visual attention mechanism with point clouds to sequentially attend to smaller regions containing the objects of interest. Bounding boxes and object categories are then estimated using a PointNet like architecture on the attended parts of the cloud, hence the name Attentional PointNet.

We claim the following contributions:

- We propose a novel deep architecture called Attentional PointNet for 3D object detection. The network directly operates on sparse 3D points, is end-to-end trainable and it learns the shape of the objects, not only their appearances from certain point of views.

- We extend visual attention mechanisms to 3D point clouds for multiple object detection. Given a cluttered environment, we show that the network learns to attend to the objects of interest, thus reducing the data needed to be processed.

- We conduct experiments on the KITTI benchmark and show that Attentional PointNet achieves near real-time performance and comparable results in LiDAR-based car detection methods.

2. Related work

With the rapid development of 3D sensor technology, LiDARs are quickly becoming a key sensor in many robotic applications. Also, the availability of many open sourced high quality annotated 3D point cloud data has motivated researchers to develop efficient feature representations to detect and localize objects in point clouds[9]. When rich and detailed point clouds are available, hand-crafted features yield satisfactory results. However, their inability to generalize and adapt to more complex shapes and unstructured environment results in limited success for autonomous navigation.

Following the general trend in computer vision, Deep Learning (DL) emerged as the dominating methodology for representation learning in point cloud processing, replacing manual feature engineering for representation of point clouds. However, In contrast to images where the detailed textured information is available, point clouds represent the outer surface of the objects in the scene. They are sparse, unordered and have a highly variable point density. Dealing with these challenges, most existing algorithms are based on the following approaches:

- Converting point clouds into 2D images [6, 10] and reinstating the state-of-art deep architectures to detect multiple objects and then projecting results back to 3D space. However, converting point clouds to 2D images results in losing essential 3D structural information of the objects. The appearance of an object from a single view can be ambiguous whereas the shape of an object would contain more information for classification and localization of the object.

- Converting point clouds into volumetric forms like voxel grids [11, 12] and generalizing image CNNs to 3D CNNs. However, for dense 3D data, computational and memory requirements grow cubically with the resolution of voxels.

- Another approach is inferring 3D bounding boxes directly from 2D images [13]. However, the depth estimation greatly affects the accuracy of image-based 3D detection.

Other work involves multi-modal fusion [14, 10, 15] combining images and LiDAR data to improve detection accuracy particularly for small objects (pedestrians, cyclists).

Recent work [3, 16, 17, 18, 19, 20] proposes novel types of network architectures, which directly process raw point clouds without converting them to other formats. Among these, PointNet [3] which is simple and works in real-time, has shown encouraging results for single object classification and semantic segmentation. We chose PointNet as the backbone for our model.

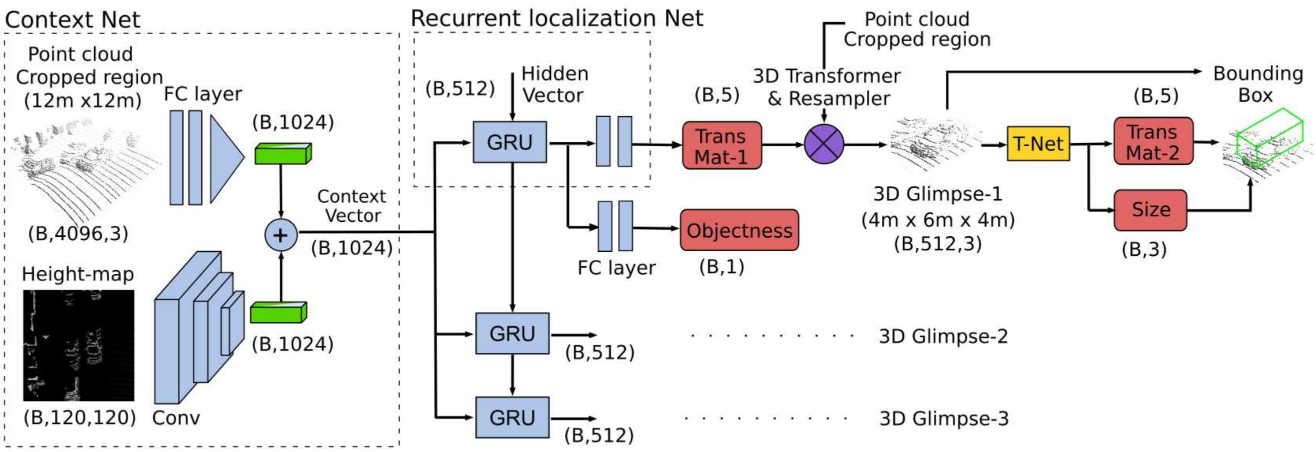

Figure 2. **Attentional PointNet for 3D object detection**: We use Attention Mechanism in 3D space to sequentially attend to relevant smaller regions (3D glimpse) of a large point cloud and classify object inside the glimpse. Given the point cloud and the corresponding height map, network sequentially regress parameters of a 3D Transformation matrix representing pose of a fixed size 3D glimpse. A modified PointNet (T-Net) then estimates another 3D transformation matrix and size representing the 3D bounding box of the object inside the glimpse. Where B is the batch size.

3. Attentional PointNet

Visual search is extensively involved in everyday perception, and biological systems like the human eye, and it manages to perform it remarkably well. As in [21], human perception does not process the whole scene in its entirety at once. Humans focus to attend to relevant parts in the scene for acquiring necessary information when and where it is needed. Focusing onto smaller relevant parts of the scene saves "computational bandwidth" as only fewer pixels need to be processed. Irrelevant parts in the scene are out of fixation and they are ignored, this reduces the complexity of the task.

Mnih *et al.* in [22, 23] proposed a deep Recurrent Neural Network (RNN) which processes a multi-resolution crop (glimpse) of input image at each iteration. Selective attention and manipulation of the data by cropping is a non- differentiable operation and the network could not be trained with backpropagation. To overcome this, Jaderberg *et al.* in [24] proposed Spatial Transformer Network (STN) module which explicitly allows the spatial manipulation of data within the network. Transformations including scaling, cropping, rotations, as well as non-rigid deformations are performed on the entire feature map (non-locally).

Recurrent-STN [25] used STN with RNN to localize and recognize multiple objects simultaneously. Bernardino *et al.* in [26] also proposed an attention mechanism based method that learns how to segment the instances sequentially. These Mechanisms have also been successfully adapted to dynamic sequences in computer vision (spatiotemporal data) [27].

Taking inspiration from human perception of sequentially recognizing the objects by moving fovea from one object to the next relevant object, and building upon the work in [24, 25, 26], we propose to use Visual Attention with point clouds in Euclidean space for the 3D object detection.

3.1. Proposed Architecture

The proposed architecture of Attentional PointNet as shown in Figure 2 consists of several core functional blocks: Context Network, Recurrent Localization Network, 3D Transformer, and Resampler, Classifier, 3D Box Estimation. The network takes the raw 3D point clouds generated from high-resolution LiDARs and outputs bounding boxes for car detection. A special loss function was designed for the network to be end to end trainable, explained in subsection 4.2.

3.2. Context Network

As its name indicates, the Context Network extracts context features of the input pointcloud, allowing it to attend to possible locations of objects of interest. It consists of two input streams: 3D Points belonging to a cropped region of $(12m \times 12m)$ and the corresponding height map, a vertical projection of the point cloud in the form of a 2D

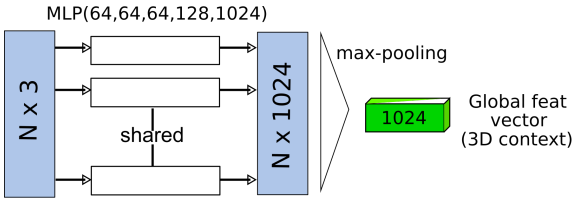

Figure 3. PointNet [3] is the backbone of our Context Network

image of size 120×120 cells. We use a simplified version of PointNet [3], where we exclude the T-Net as shown in Figure 3. Through a series of 1D convolutions (Multi-Layer Perceptron), the networks converts the point set into a higher dimensional feature space, followed by max pooling as a symmetric function to aggregating information from all the points. The resulting vector is denoted as 3D context.

Aforementioned height map is generated by projecting the point set on the ground plane, details of which are given in subsection 4.1. We use standard convolutions followed by max pooling layer to convert it into a feature vector, denoted as 2D context.

Finally, we fuse 3D context with 2D context by summing them, projecting the representations into a joint space. Alignment of two embedding spaces was crucial after thorough experimentation we found addition gives equally good results as concatenation while keeping the network capacity low.

The two contexts are complementary. The bird's eye view alone is not discriminative enough to differentiate structures that look similar to a car when projected vertically, while the 3D context in such cases can provide useful insights; On the other hand, understanding complex structures, such as bushes, is difficult in 3D. In this case, having a 2D perspective of the data clearly provides complementary information.

3.3. Recurrent Localization Network

The Recurrent Localization Network is the heart of our Attentional PointNet architecture. Unlike in [4] [5], where 2D RGB images are used to detect objects and then are projected into 3D space to obtain the region proposals, we aim for a LiDAR only solution. Inspired by [26] [25], the recurrent localization network sequentially attends to the location of the new object at every iteration (i). This module consist of two parts, as illustrated in Figure 2:

(i) The recurrent part consist of a GRU layer, which takes the context vector C of size (B, 1024) as input from the context network and h_{i-1} a hidden vector from the GRU cell in previous iteration $(i-1)$. It outputs a vector h_i of shape $(B, 512)$ which is the input to the Localization Network.

(ii) The localization part is a 3 layer fully connected network which takes h_i as input from the GRU cell at every iteration and, similar to [24], regresses 5 parameters $(\cos\theta_i, \sin\theta_i, Tx_i, Ty_i, Tz_i) \in \Theta_i$ of a 3D transformation matrix, which corresponds to the attention operation (selecting the attended glimpse). Let $\Theta = \{\Theta_1, \Theta_2 ... \Theta_n\}$ be the set of the transformation matrix parameters at each iteration, whereas n is the number of iterations.

For simplicity we have only considered rotation along z-axis and Transformation matrix can be written as:

$$T(\Theta_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & Tx_i \\ \sin\theta_i & \cos\theta_i & 0 & Ty_i \\ 0 & 0 & 1 & Tz_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where θ is the angle of rotation along the z-axis. $\theta_i = tan^{-1}(\sin\theta_i / \cos\theta_i)$. We only consider the rigid 3D transformations and neglect the scale and shear. This is evident in the case of pointcloud and in contrast with images, as the scale/ size of the object does not change with respect to the distance of the object from the sensor. Unlike the original STN [24] that has no direct supervision on transformation $T(\Theta_i)$, we explicitly supervise our localization network to predict object locations.

This can be formulated as follows:

$$C = f_{context}(I) \tag{2}$$

$$h_i = f_{RNN}(C, h_{i-1}) \tag{3}$$

$$T(\Theta_i) = f_{loc}(h_i) \tag{4}$$

where $f_{context}$ is the context network taking I as input (point set and height-map) and outputs the context vector C, f_{RNN} is a GRU cell, and f_{loc} is the Localization Network. Here, a rigid transformation $T(\Theta_i)$ is produced at each time-step from the hidden state of the RNN. Importantly, the rigid transformations are conditioned on the previous transformations through the time dependency of the RNN.

3.4. 3D Transformer and Resampler

To make the attention operation differentiable, and the whole network trainable end-to-end, we resort to a 3D Transformer network. It takes the transformation matrix parameters as input and transforms the input pointcloud $P(4096, 3) \rightarrow P'(4096, 3)$. The pointwise rigid 3D transformation is given by:

$$\begin{bmatrix} x_i^t \\ y_i^t \\ z_i^t \\ 1 \end{bmatrix} = T(\Theta_i) \begin{bmatrix} x_i^s \\ y_i^s \\ z_i^s \\ 1 \end{bmatrix} \tag{5}$$

where (x_i^t, y_i^t, z_i^t) are the transformed coordinates of output pointcloud P', $(x_i^s; y_i^s, z_i^s)$ are the source coordinates of the input pointcloud P, and $T(\Theta_i)$ is the rigid transformation matrix.

Let the input point cloud be in the bounding box of size (W, L, H) centered at $(0, 0, 0)$ in $\mathbb{R}^3$ space is transformed such that the points belonging to object of interest fall inside a smaller bounding box of size (W', L', H') centered at $(0, 0, 0)$ in $\mathbb{R}^3$ space. This is more clearly illustrated in Figure 4.

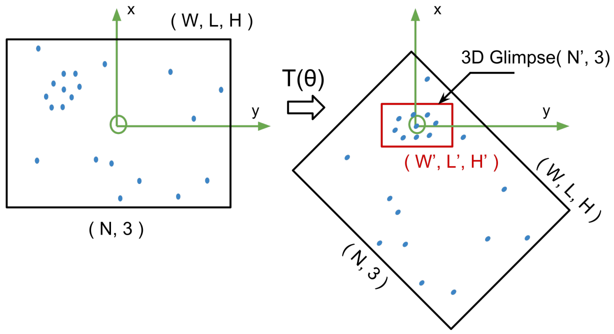

Figure 4. 2D illustration of working of 3D Transformer

As the network attends to the points falling inside the smaller bounding box, it can be called as a 3D Attention as shown in Figure 2. Points inside 3D glimpse are cropped and are resampled with replacement to 512 points.

3.5. Localization and recognition

Given the points inside the attended region (3D glimpse), this module estimates an oriented 3D bounding box of the object. For this purpose, we use a light-weight regression PointNet(T-Net) as in [4]. Our modified T-Net regresses 5 parameters $(\cos\delta_i, \sin\delta_i, tx_i, ty_i, tz_i) \in \Delta_i$ of 3D transformation matrix representing true centre and orientation of the object and it also regresses 3 parameters (H, W, L) of size of the 3D bounding box. Let $\Delta = \{\Delta_1, \Delta_2..\Delta_n\}$ be the set of transformation matrix parameters representing the bounding box of object inside the 3D glimpse. The final bounding box location and orientation can be found by:

$$T(\Psi_i) = T(\Theta_i) * T(\Delta_i) \quad (6)$$

The objects are classified with a 2 layer fully connected network which takes h_i as input from the GRU cell at every iteration and outputs a score (objectness) indicating the probability of having a specified object in the attended region.

4. Training & Experiments

We evaluate our model on the KITTI 3D object detection benchmark [28] which contains 7,481 training images/ point clouds and 7,518 test images/point clouds. In this work, we only evaluate our network on the *Car* category. Since the KITTI dataset does not provide ground truth for the test set and the access to the test server is limited, we use a similar evaluation protocol as the one used in [14], [7]. We split the training dataset into 70/30 percent ratio as a training set and a validation set respectively. The split avoids samples from the same sequence being included in both the training and the validation set.

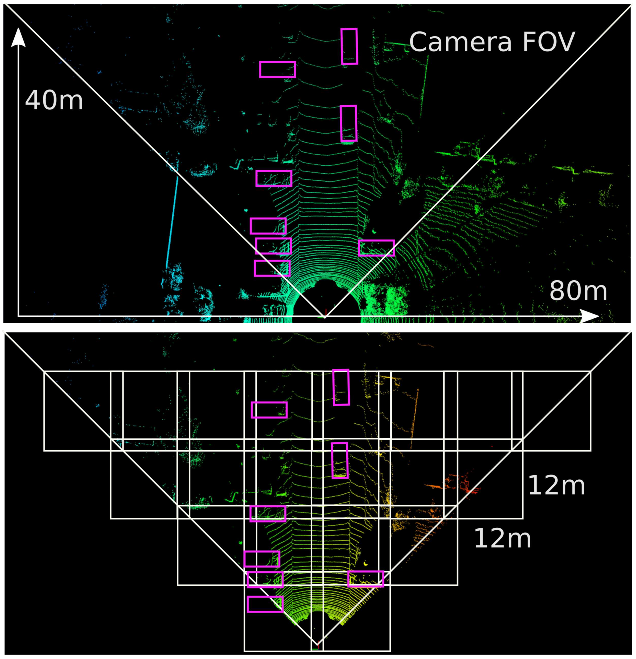

Figure 5. *(a)* Illustrates the expanse of FOV of camera in the birds-eye-view with sample labels. *(b)* Shows subdivision of FOV area into equally spaced cropped regions of (12m x 12m).

4.1. Data Augmentation

The point cloud data in each frame of KITTI dataset is typically composed of $\approx 100k$ points and has a range of about 120m. As in this paper, we aim to directly work with the point set, using the whole point cloud data as input to the network is impractical. Moreover, the KITTI dataset only provides labels of the objects in the field of view (FOV) of the camera [28] as shown in Figure 5. We therefore remove all points falling out of the FOV.

We train the model on a custom dataset, which was generated by augmenting the KITTI dataset. To this end, we subdivide the FOV area from each scan into equally spaced cropped regions of $12m{\times}12m$ with an overlap of 1m as shown in Figure 5. We illustrate the effect of the 3D Visual Attention Mechanism on these cropped regions.

Each cropped region consists of a number of points ranging between 20,000 to none. Directly processing all the points not only imposes increased memory/efficiency burdens on the computing platform; the highly variable point density throughout the space might also bias the detection. We randomly sample each cropped region to a fixed number of N=4096 points.

Inspired by [14], [6], each cropped region of size $12m{\times}12m$ is also converted into a grayscale image of size $120{\times}120$ pixels encoding height information as shown in Figure 6. We projected and discretized the 3D point clouds into a 2D grid with resolution of about r=$10cm$. We choose $z \in [-2m, 3m]$, to cover an area above the ground to

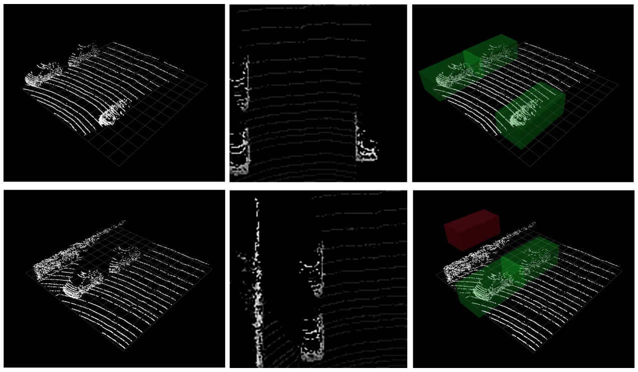

Figure 6. Left: point cloud of 12m x 12m cropped region. Center: Corresponding height-map. Right: Associated sequence of 3 labels (bounding boxes), second row consist of only two cars (green) hence one of the bounding box (red) is outside the cropped region.

about 3m height, expecting trucks as highest objects. Let $\mathrm{P} \in \mathbb{R}^3$ be the set of all the points in a cropped region and $H \in \mathbb{R}^{mxm}$ be the height map with $m = 120$. We define $\S_j = f_{\mathrm{PS}}(\mathrm{P}_i, r)$ with $S \in \mathbb{R}^{mxm}$ mapping each point with index i in cropped region into a grid cell S_j of our height-map. The set describing all points mapped into a a specific grid cell of height map can be defined as:

$$\mathrm{P}_{i \to j} = \{\mathrm{P}_i = [x, y, z]^T | S_j = f_{\mathrm{PS}}(\mathrm{P}_i, r)\} \tag{7}$$

$$H(S_j) = max(\mathrm{P}_{i \to j}.[0, 0, 1]^T) \tag{8}$$

Next, we generate a sequence of labels for each cropped region. The information from KITTI dataset is used to check if there are any cars inside each cropped region and we note the position, orientation and size of them (green). After a thorough analysis, we found around 95% of all the cropped regions of 12m x 12m area have a maximum of 3 cars. So, for each cropped region we have a sequence (i) of 3 labels $\Psi^{gt} = \{\Psi_1^{gt}, \Psi_2^{gt}, \Psi_3^{gt}\}$ the set of transformation matrix parameters representing the ground-truth bounding boxes where $\Psi_i^{gt} = (\cos\psi_i^{gt}, \cos\psi_i^{gt}, Tx_i^{gt}, Ty_i^{gt}, Tx_i^{gt})$. If there are less than 3 cars, we generate bounding boxes of a fixed size at a fixed location outside the cropped region as non-car (red). Figure 6 shows some samples of our generated dataset.

The KITTI dataset consists only a few numbers of ground truths in each frame which lead to a significantly higher number of cropped regions without cars than those with the cars. In order to have a balanced dataset, authors in SECOND [8] have used a data augmentation approach to first, generate a database containing the labels of all ground truths and their associated point cloud data (points inside the 3D bounding boxes of the ground truths), then randomly introducing several ground truths from this database into current training point cloud via concatenation. We have opted a simpler approach and kept the number of cropped regions with no cars and with cars in equal proportion. Our augmented KITTI dataset consists of a total of 27,041 cropped regions for training.

4.2. Loss Function

We jointly train the full model including all modules (context network, recurrent localization Network, classifier, and 3D box estimation T-Net) with the following set of losses:

$$L_{seq-i} = \alpha * L_{cls} + \beta(L_{T1-reg} + L_{T2-reg}) \\ + \gamma * L_{size-reg} + \lambda * L_{reg} \tag{9}$$

$$L_{final} = \frac{1}{3}\sum_{i=1}^{n=3} L_{seq-i} \tag{10}$$

$$L_{reg} = ||I - T(\Psi)T(\Psi)^T||^2 \tag{11}$$

where L_{seq-i} is the total loss for a sequence, L_{cls} is the classification loss, L_{T1-reg} is for the transformation matrix parameters regressed by the Localization Network, L_{T2-reg} is for the transformation matrix parameters regressed by T-Net and $L_{size-reg}$ is the regression loss for bounding box size. We used binary cross-entropy loss for the classification task and smooth-l1 (Huber) loss is used for all the regression cases. L_{reg} is the regularization loss. We constrain our predicted transformation matrix to be close to the orthogonal matrix. As in [3], it helps the optimization to become more stable and the network achieves better performance. For training, we have predefined the length of the predicted sequence as n=3. A primary challenge of the sequential detection is matching predictions and ground-truth instances. We compute a maximum-weighted bipartite graph matching between the output instances and ground-truth instances as in [26], [29]. Matching makes the loss insensitive to the ordering of the ground-truth instances. The matching weight M_{ij} is the IoU score between a pair of detections and the ground-truth instances. We use the Hungarian algorithm to compute the matching; we do not back-propagate the network gradients through this algorithm.

$$M_{ij} = f_{iou}(\Psi - \Psi^{gt}) \tag{12}$$

$$\Psi_{matched} = f_{match}(M_{ij}) \tag{13}$$

The network performs three predictions for each input cropped region. At each sequence, the network focuses on the new object. If there are less than three objects, we explicitly force the network to focus on the outside of the cropped region and classify as negative detection for the remaining number of sequences.

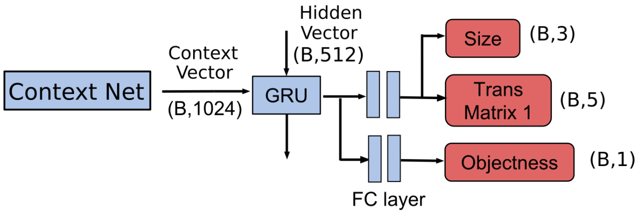

Figure 7. Attentional PointNet (vanilla) network architecture: Localization Network directly regress the size of bounding box.

4.3. Network and Training Details

The 3D Context Network consists of three fully connected layers implemented as 1D convolutions with input-output feature sizes as $(3, 64), (64, 128), (128, 1024)$ respectively for each layer. For each input point, we only use x, y, z coordinate values as attributes. All the layers include ReLU and batch normalization except for the first layer which does not have Batch normalization layer. Localization Network also consists of three fully connected layers with input-output sizes as $(512, 256), (256, 128), (128, 5)$ respectively. Only the first two layers include ReLU and only the first layer includes batch normalization. The transformation matrices are firstly initialized by the identity matrix. In the loss function, the values of the hyperparameters are kept as $\alpha = 1, \beta = 1.5, \gamma = 0.5$ and $\lambda = 0.01$.

We train the model with stochastic gradient descent (SGD) with a momentum of 0.9, weight decay of 0.0005 and a batch size (B) of 32. We keep the learning rate to be 0.01 for the first 40 epochs and then lower it to 0.001 for further epochs. We observed that the network converges in $\approx$ 120 epochs. Training on our custom KITTI dataset takes 8 to 9 hours to converge with PyTorch and GTX 1080GPU.

To evaluate the effectiveness of our 3D bounding box estimation module, we also trained a vanilla version of our Attentional PointNet network as shown in Figure 7. In the vanilla version, we remove the 3D bounding box estimation module and directly regress the sizes from the localization network and use Θ transformation matrix parameters for box center coordinates and orientation.

Hardware — We trained, validated and tested the model on an Intel Xeon CPU W3520 and a GeForce GTX 1080 GPU with 8GB on Ubuntu 16.04 using PyTorch version 1.0.

5. Results

3D detection is a more challenging task as it requires finer localization of objects in 3D space. For the Car category, we compare the proposed method with several top-performing algorithms, including multi-sensor approaches: MV3D [14], Frustum PointNet [4] and RoarNet [5] ; LiDAR based approaches: VeloFCN [30], VoxelNet [7] and RT3D [31]; LiDAR based birds-eye-view (BV) approach: Complex-YOLO [6];. We train Attentional-PointNet from scratch using only the LiDAR data provided in KITTI dataset. Finally we provide our results as Average Precision (for $IoU > 0.7$) on KITTI dataset for the detection of Car category.

Table 1. Performance comparison in 3D detection: average precision (in %) on KITTI validation set. Note that our method is validated on our splitted validation dataset, whereas all others are validated on the official KITTI test set. All values are from the official KITTI leaderboard.

Method	Modality	FPS	Car		
			Easy	Mod.	Hard
MV3D [14]	Lidar+Mono	2.8	71.09	62.35	55.12
F-PointNet [4]	Lidar+Mono	5.9	81.20	70.39	62.19
AVOD [32]	Lidar+Mono	12.5	73.59	65.78	58.38
RoarNet [5]	Lidar+Mono	10	83.95	75.79	67.88
VeloFCN [30]	Lidar	-	15.20	13.66	15.98
RT3D [31]	Lidar	11.23	23.49	21.27	19.81
VoxelNet [7]	Lidar	4.3	67.27	52.87	46.62
Complex-YOLO [6]	Lidar (BV)	16.6	55.63	49.44	44.13
A-PointNet (vanilla)	Lidar	12.5	49.47	44.64	41.71
Attentional-PointNet	Lidar	8.06	58.62	52.28	47.23

Table 2. Analysis of computation time required by Attentional PointNet

Task	Time
Pre - processing	0.084 sec
Model forward pass	0.038 sec
post-processing & NMS	0.002 sec
Total	0.124 sec

Table 1 summarizes the comparison for the Car category detection, Attentional PointNet achieves comparable Average Precision of 52.28% for moderate difficulty among the network architectures using LiDAR data only. Also, in terms of inference time, Attentional PointNet shows credible performance. The vanilla version outperforms all the approaches except Complex-YOLO which only uses the birds-eye-view projection of LiDAR data, the computation time is lower but it suffers in 3D detection accuracy. We can observe that the multi-sensor approaches achieve significantly higher accuracy compared to those using LiDAR data only.

To process 20 cropped regions of 12m x 12m as shown in Figure 5 the inference time is 124ms. Network outputs bounding boxes at 8Hz with GTX 1080 GPU and hardware specified further above. From Table 2 it can be observed that pre-processing, which involves cropping the point cloud to a set of 20 regions of size $12m{\times}12m$ and converting them into height-map, is the computationally most expensive task. Currently, we use Python and Numpy for the Pre-processing tasks but parallelizing them on GPU would greatly improve the inference time and is planned as future work.

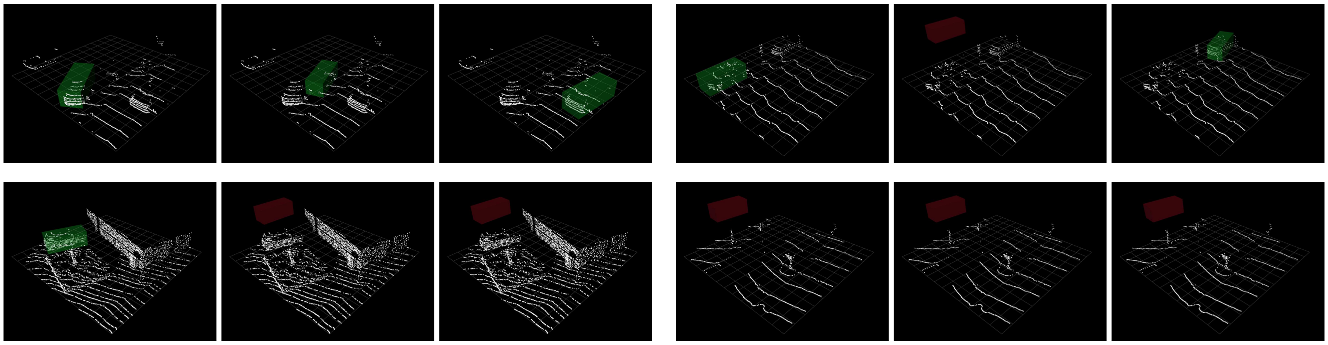

Figure 8. Results on $12m \times 12m$ cropped regions extracted from the KITTI dataset. For each cropped region network makes three predictions sequentially classifying and localizing the cars in the scene. It can be observed that the network is effective and capable of attending/finding multiple Cars even in a highly cluttered environment. When there are less than three cars in the scene, the network focuses outside the cropped region for the remaining number of predictions and appropriately classify them as negative detections.

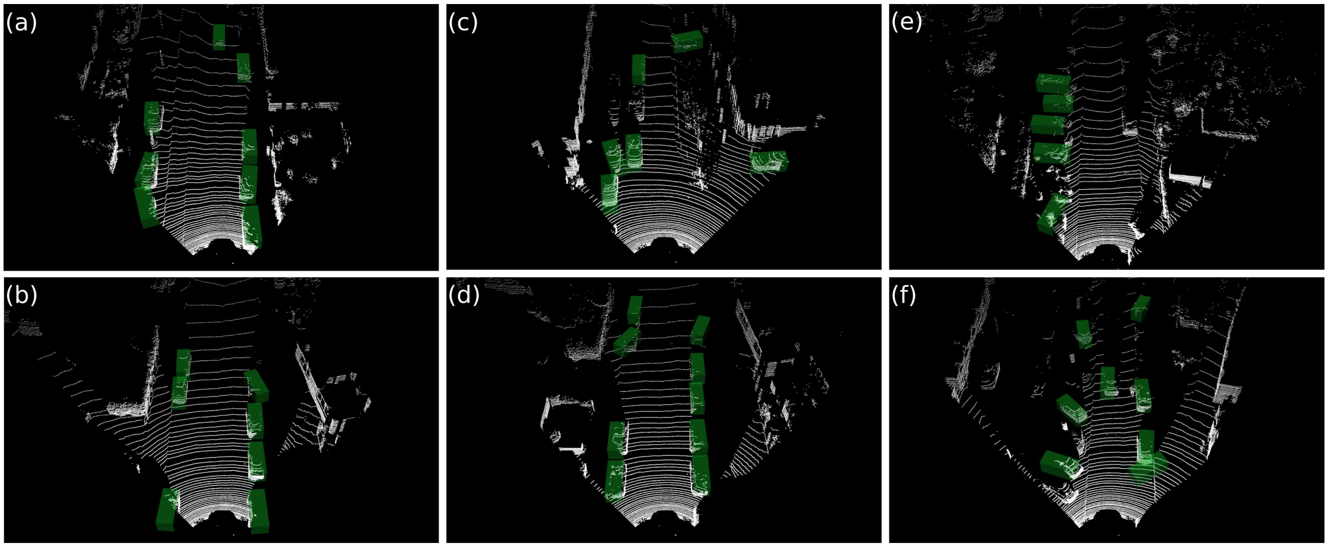

Figure 9. Visualizations of **Attentional PointNet** results on KITTI dataset for the car category. (a),(b): our model's ability to detect multiple objects in cluttered environments; (c): successful discrimination between a car and van; (e),(f): some failure cases with false positive detections and the orientation of the cars not accurately estimated.

6. Conclusions

Most existing methods for 3D object detection in large scale point clouds either rely on hand-crafted feature representations or multi-sensor approaches. In this work, we present a novel end to end trainable deep architecture Attentional-PointNet for 3D object detection in the point cloud. The network only uses 3D data from LiDAR and captures 3D geometric information of the data effectively. To reduce the search space for object detection we proposed to use Attention Mechanism with 3D point clouds and introduce a new recurrent 3D Localization Network module. We conducted experiments with the KITTI dataset and evaluated our results for car category detection. We demonstrate our network's capability to sequentially attend/focus on the new object in each iteration. For car detection on KITTI dataset, Attentional PointNet shows comparable results with existing state-of-the-art LiDAR-based 3D detection methods and surpasses many approaches in terms of inference time.

7. Acknowledgements

This work was conducted at Inria, team Chroma. The Authors want to thank all the members of the team for constant support throughout the process of research and writing of this work. This work has been conducted within the ENABLE-S3 project that has received funding from the ECSEL joint undertaking under grant agreement NO 692455. The work was also partially funded by ANR grant Deepvision (ANR-15-CE23-0029, STPGP-479356-15).

References

[1] Alberto Broggi, Alex Zelinsky, Umit Ozguner, and Christian Laugier. Handbook of Robotics 2nd edition, Chapter 62

on "Intelligent Vehicles". In Bruno Siciliano and Oussama Khatib, editors, *Handbook of Robotics 2nd Edition*, pages 1627–1656. Springer, July 2016. 1

[2] Yong-Joo Oh and Yoshio Watanabe. Development of small robot for home floor cleaning. In *SICE 2002. Proceedings of the 41st SICE Annual Conference*, volume 5, pages 3222–3223. IEEE, 2002. 1

[3] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 1, 2, 3, 4, 6

[4] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 4, 5, 7

[5] Kiwoo Shin, Youngwook Paul Kwon, and Masayoshi Tomizuka. Roarnet: A robust 3d object detection based on region approximation refinement. *arXiv preprint arXiv:1811.03818*, 2018. 2, 4, 7

[6] Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross. Complex-yolo: Real-time 3d object detection on point clouds. *arXiv preprint arXiv:1803.06199*, 2018. 2, 5, 7

[7] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 2, 5, 7

[8] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2, 6

[9] Muhammad Zeeshan Zia, Michael Stark, and Konrad Schindler. Are cars just 3d boxes?-jointly estimating the 3d shape of multiple objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3678–3685, 2014. 2

[10] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018. 2

[11] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1355–1361. IEEE, 2017. 2

[12] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015. 2

[13] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016. 2

[14] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *IEEE CVPR*, volume 1, page 3, 2017. 2, 5, 7

[15] Özgur Erkent, Christian Wolf, Christian Laugier, David Sierra Gonzalez, and Victor Romero-Cano. Semantic grid estimation with a hybrid bayesian and deep neural network approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018. 2

[16] 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*, pages 5099–5108, 2017. 2

[17] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018. 2

[18] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018. 2

[19] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. Exploring spatial context for 3d semantic segmentation of point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–724, 2017. 2

[20] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2598–2606, 2018. 2

[21] John K Tsotsos. Analyzing vision at the complexity level. *Behavioral and brain sciences*, 13(3):423–445, 1990. 3

[22] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014. 3

[23] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014. 3

[24] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 3, 4

[25] Søren Kaae Sønderby, Casper Kaae Sønderby, Lars Maaløe, and Ole Winther. Recurrent spatial transformer networks. *arXiv preprint arXiv:1509.05329*, 2015. 3, 4

[26] Bernardino Romera-Paredes and Philip Hilaire Sean Torr. Recurrent instance segmentation. In *European conference on computer vision*, pages 312–329. Springer, 2016. 3, 4, 6

[27] Fabien Baradel, Christian Wolf, Julien Mille, and Graham W Taylor. Glimpse clouds: Human activity recognition from unstructured feature points. *Computer Vision and Pattern Recognition (CVPR)*, 3, 2018. 3

[28] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 5

[29] Mengye Ren and Richard S Zemel. End-to-end instance segmentation with recurrent attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6656–6664, 2017. 6

[30] B. Li. 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017. 7

[31] Yiming Zeng, Yu Hu, Shice Liu, Jing Ye, Yinhe Han, Xiaowei Li, and Ninghui Sun. Rt3d: Real-time 3-d vehicle detection in lidar point cloud for autonomous driving. *IEEE Robotics and Automation Letters*, 3(4):3434–3440, 2018. 7

[32] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018. 7