ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA

DIPARTIMENTO di
INGEGNERIA DELL'ENERGIA ELETTRICA E DELL'INFORMAZIONE
"Guglielmo Marconi"
DEI

CORSO DI LAUREA MAGISTRALE IN
Advanced Automotive Electronic Engineering

TESI DI LAUREA
in
Neural Network Computing, A.I. and Machine Learning for Automotive

3D StixelNet
Deep Neural Network for 3D object detection stixel-based

CANDIDATO

Capuzzo Davide

RELATORE

Chiar.mo Prof. Riccardo Rovatti

CORRELATORE

Dott. Lorenzo Baraldi

Stefano De Battisti
Andrea Zinelli
Riccardo Maiolani

Anno Accademico
2019/2020

Sessione
II

Sommario

In questa tesi è stato presentato un algoritmo di deep learning per il rilevamento di oggetti 3D da nuvola di punti in ambiente esterno. Questo algoritmo è alimentato con stixel, un dato di tipo medio generato partendo da una nuvola di punti o da una mappa di profondità. Uno stixel può essere pensato come un piccolo rettangolo che inizia dalla base della strada e poi sale fino alla sommità dell'ostacolo che riassume la superficie verticale di un oggetto. L'obiettivo di stixel è comprimere i dati provenienti dai sensori in modo da avere una trasmissione veloce senza perdere informazioni.

L'algoritmo per generare stixel è un nuovo algoritmo da me sviluppato che può essere applicato sia dalla nuvola di punti generata dal LIDAR che dalla mappa di profondità generata dalla camera stereo e mono.

I passaggi principali per creare questo tipo di dati sono:

- l'eliminazione dei punti che giacevano sul piano stradale;
- la creazione di una matrice che riassuma la profondità di gruppo degli stixel;
- la creazione di stixel unendo tutte le celle che fanno parte dello stesso oggetto.

La generazione di stixel riduce i punti da 40.000 a 1200 per la nuvola di punti LIDAR e da 480.000 a 1200 per la mappa di profondità.

Per estrarre informazioni 3D dallo stixel, questi dati sono stati inseriti in un algoritmo di deep learning adattato a ricevere in input questo tipo di dati. L'adattamento è stato effettuato partendo da una rete neurale esistente per il

rilevamento di oggetti 3D in ambiente indoor. Questa rete è stata adattata per superare la scarsità di dati e le grandi dimensioni della scena.

Nonostante la riduzione del numero di dati, grazie alla giusta messa a punto, la rete creata in questa tesi ha potuto raggiungere lo stato dell'arte per il rilevamento di oggetti 3D.

Questo è un risultato rilevante perché apre la strada all'utilizzo di dati di tipo medio e sottolinea che la riduzione dei punti non significa una riduzione delle informazioni se i dati vengono compressi in modo ottimale.

Abstract

In this thesis it has been presented an algorithm of deep learning for 3D object detection from the point cloud in an outdoor environment. This algorithm is feed with stixel, a medium-type data generates starting from a point cloud or depth map. A stixel can be think as a small rectangle that start form the base of the road and then rises until the top of the obstacle summarizing the vertical surface of an object. The goal of stixel is to compress the data coming from sensors in order to have a fast transmission without losing information.

The algorithm to generate stixel is a novel algorithm developed by myself that is able to be applied both from point cloud generated by lidar and also from depth map generated by stereo and mono camera.

The main passage to create this type of data are:

- the elimination of points that lied on ground plane;
- the creation of an average matrix that summarizes the depth of group of stixel;
- the creation of stixel merging all the cells that are of the same object.

The stixel generates reduce the points from 40 000 to 1200 for LIDAR point cloud and to 480 000 to 1200 for depth map.

In order to extract 3D information from stixel this data has been feed into a deep learning algorithm adapted to receive as input this type of data. The adaptation has been made starting from an existing neural network use for 3D object detection in an indoor environment. This network has been adapted in order to overcome the sparsity of data and to the big size of the scene.

Despite the reduction of the number of data, thanks to the right tuning the network created in this thesis have been able to achieve the state of the art for 3D object detection.

This is a relevant result because it opens the way to the use of medium-type data and underlines that the reduction of points does not mean a reduction of information if the data are compressed in a smart way. oints not means a reduction of information if the data are compressed in a smart way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cars are becoming more and more smarting. The introduction of new sensors and new electronic systems is a standard in the automotive world. The final goal of this implementation is to create a driverless car in order to increase the safety of the drivers and other occupants of the road, reduce the travel timing and let people be free to apply the time travel to other interests.

The task that an autonomous car has to do at the same time in order to provide a reliable service are:

- Perception: The perception is the analysis of the environment understanding the whole scene and analyzing the critical situations.
- Planning: the choice of the right path both from a high-level abstraction like which road take in order to arrive at the destination and also from a low level so which trajectory follow taking into account the environment.
- Control: Take the information provide by the low-level planning and decide how to set the engine, the steering wheel, or other actuators in order to follow the best trajectory.

This set of high-level service is provided in a reduced way also in the car that is possible to buy nowadays.

In the automotive world, there are many levels of autonomy of a vehicle defined by the SAE (Society of Automotive Engineers). These levels are divide by which task a car has to do and which autonomy a car has. They are:

- Level 0: No Automation

  Level 0 is the base level where all aspects of driving being fully human and manually controlled.

- Level 1: Driver Assistance

  Level 1 is the lowest level of automation. Only one single aspect of driving is automated. In particular, the aspect that is automated is either steering or acceleration/deceleration.

- Level 2: Partial Automation

  In level 2 the vehicle is able to control both the steering and acceleration/ deceleration ADAS capabilities. At this level, the driver has complete control of the vehicle at all times. Examples of level 2 include lane-keeping, adaptive cruise control and self-parking features.

- Level 3: Conditional Automation

  In level 3 a vehicle is able to detect the environment around it. level 3 vehicles contain the lowest-tier system that is classified as an automated driving system as opposed to a manual system. With this more advanced technology, level 3 vehicles can make informed decisions for themselves such as overtaking slower-moving vehicles. However, with the expectation that the human driver will respond appropriately to a request to intervene in a hard task or system failure.

- Level 4: High Automation

  In level 4 vehicles, even if a human driver does not respond appropriately to a request to intervene car can pull over safely by the guiding system. In this sense, these cars are left completely to their own devices without

any human intervention in the vast majority of situations. The option to manually override does remain in difficult or preferable circumstances.

- Level 5: Full Automation

  In level 5, human driving is completely eliminated. A level 5 vehicle must have a perfect response to all the situations include off-road driving and other terrains that Level 4 vehicles may not necessarily be able to detect or intelligently comprehend. In other words, level 5 vehicles have a much more advanced environment detection system.

  In this class of automated vehicles, there aren't the typical driving controls such as steering wheels, gas and brake pedals, or others. At this level, there is not the possibility of intervention from a human driver.

From level 2, the perception task is becoming relevant. And in order to have a good understanding of the environdìment, a car needs to know not only the roads and the generic obstacle but also understand the position and type of occupants of the road. In order to achieve this task, many options and many paths have been taken.

The most important way to understand the occupancy of the road is taking the input from some sensor like LIDAR or stereo camera and elaborate this input with a deep learning algorithm.

A deep learning algorithm is a set of operation that takes an input some data and apply some operations like multiplication for some weights and discretization functions like sigmoid in order to extract some information independently. The weights of the multiplication are learned in a phase called training where taking as input a annotate data set the algorithm tune this weight in order to provide the right output. The differentiation of each network is provided by the sequence of operations that, after many trials and consideration, have been understood that is more suitable for each task.

After this training and tuning phase, each network is ready to do a specific task.

The goal of this thesis is to create a deep learning algorithm for 3D object detection for an autonomous vehicle using a medium type of data that has been created starting from a point cloud or depth map generated from the stereo and mono camera.

The dataset used for the training of the Neural Network is the Kitti dataset that will be illustrated in the following paragraphs.

In the second chapter, it has been illustrated all the types of deep learning algorithms which has been inspired the algorithm developed in this thesis.

The third chapter, it is shown the method of transformation of the point cloud to medium type data that will be feed into the network.

The fourth chapter there is explained in detail the architecture of the network adopted to extract the 3D object detection.

The fifth chapter there is illustrated the result and the trial of the network training.

The sixth chapter is showed the conclusion and final considerations of the thesis.

1.1 Sensors

In order to have a perception of the environment a key role is provided by the sensors. A sensor suitable for 3D object detection should be reliable in all the situation. It should have enough accuracy in order to provide a resolution suitable for capturing the detail for 3D object detection. Other important things, it should have also a sufficient range that allows having enough vision of the scene. In order to achieve all this task, the sensors selected for 3D object detection are stereo Cameras and LIDARs.

1.1.1 Stereo camera

The stereo camera is sensors that use a pair of the camera to extract a depth map through some algorithms that can be both deep learning algorithms or computer vision algorithms.

Figure 1.1: Example of stereo camera

The concept is that knowing the position of two cameras and the potion along the same axis of a camera of an object is possible to understand the distance camera-object with a simple trigonometrical formula.

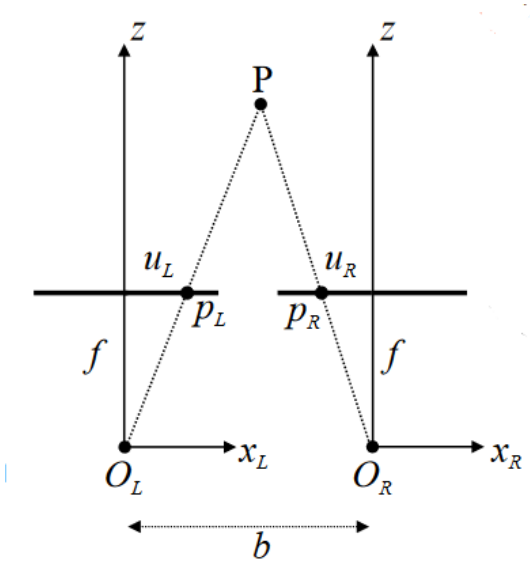


Figure 1.2: Stereo Camera Model

In Figure 1.2 it has been illustrated the working principle of a stereo camera. In this figure:

- b represent the base distance between the two cameras focal centers;
- f represent the focal distance of the two cameras;

- u_L and u_R represent the object P in the camera reference frame;

In order to evaluate the distance of the point P the procedure is:

1. find the disparity $d = u_L - u_R$
2. evaluate the distance $z = \frac{b \times f}{d}$

The real problem is to match each point in the left image with the one in the right image in order to find the disparity. There are many ways to do that. The most common is using a scanning algorithm but is slow and imprecise or use deep learning to extract disparity maps.

The advantages of using cameras to evaluate depth map are:

- cameras are cheap sensors;
- cameras can achieve high resolution and high point density;
- cameras are mature technology.

The drawback is:

- the depth map obtained from camera nowadays is not so precise and needs some computation to be extracted.

1.1.2 LIDAR

LIDAR is acronyms for Light Detection and Ranging. This sensor sends a Laser pulse train, which is sent to the surface/target to measure the time and it takes to return to its source.

The actual calculation for measuring how far a returning light photon has traveled to and from an object is calculated by:

$$\text{Distance} = \frac{\text{Speed of Light} \times \text{Time of Flight}}{2}$$

Figure 1.3: Example of LIDAR sensor

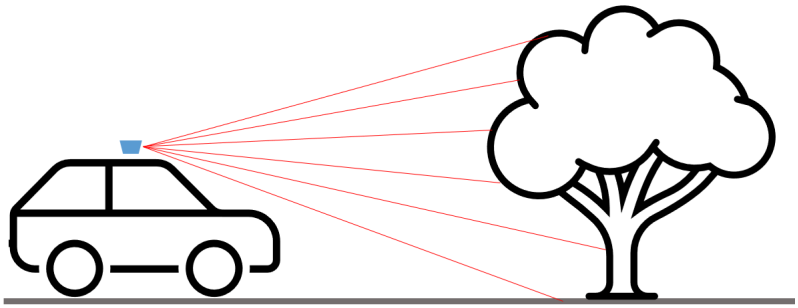


Figure 1.4: LIDAR 2D representation

In an autonomous car, the LIDAR is composed of a sensor that sends up to 8 rays and rotates by 360 degrees. These 8 rays rotating can capture the distance of 8 points and then create a point cloud.

The advantage of using a LIDAR are:

- high precision and accuracy;
- less computation to extract the 3D point cloud.

The disadvantages of this technology are:

- Not a mature technology;
- High cost;
- Less resolution;
- Less robustness to vibration.

These draw-backs are referred to mechanical LIDAR where the rotation of the laser is provided by mechanical motors. Nowadays new technology is going to be developed like solid-state LIDAR that will provide high accuracy, less cost, and high resolution.

1.2 The Kitti Dataset

The evaluation of 3D pose estimation of the vehicle is provided by a deep learning algorithm that in order to be trained needs a robust dataset. The dataset used in this thesis is Kitti dataset [1] [2], one of the most important dataset use for research in autonomous driving.

This dataset has been created by *Karlsruhe Institute of Technology* recording some driving around a mid-size city, in rural areas, and on highways.

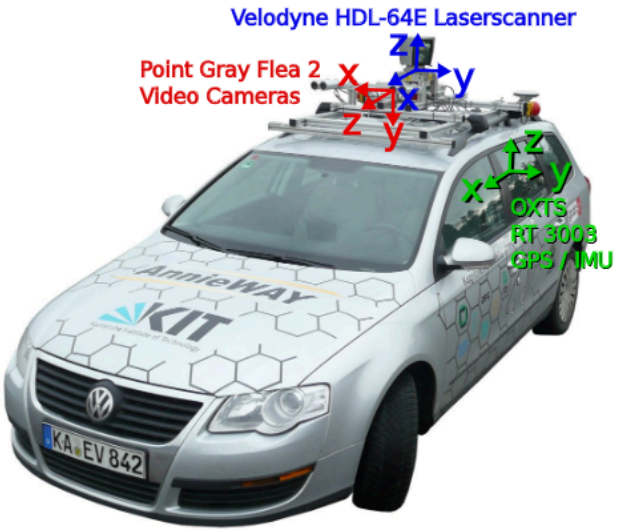


Figure 1.5: Kitti recording platform

The recording platform is a Volkswagen station wagon equipped with two high-resolution stereo camera systems (grayscale and color), a Velodyne HDL-64E laser scanner that produces more than one million 3D points per second, an OXTS RT 3003 localization system which combines GPS, GLONASS, an IMU, and RTK correction signals. The cameras, laser scanners, and localization systems are calibrated and synchronized, providing accurate ground truth.

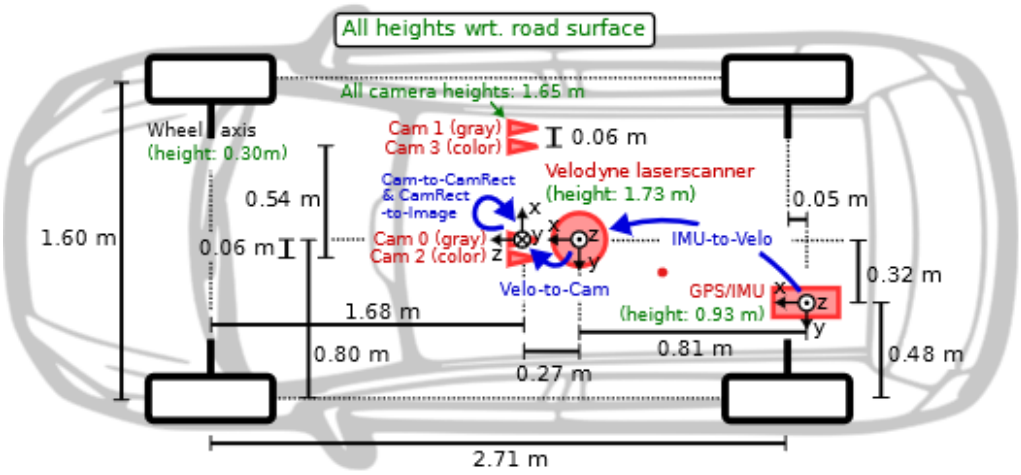


Figure 1.6: Kitti recording platform configuration

In figure 1.6 it has been illustrating the dimensions and mounting positions of the sensors (red) with respect to the vehicle body. Heights above ground are marked in green and measured with respect to the road surface. Transformations between sensors are shown in blue.

In total, the dataset contains 6 hours of traffic scenarios. The scenarios are diverse, capturing real-world traffic situations, and range from freeways over rural areas to inner-city scenes with many static and dynamic objects.

For each dynamic object within the reference camera's field of view, it has provided annotations in the form of 3D bounding box tracklets, represented in Velodyne coordinates. For each bounding box, it is assigned class and its 3D size (height, width, length) and 3D orientation on roll pitch and yaw angles.

The classes of this objects are: `"Car"`, `"Van"`, `"Truck"`, `"Pedestrian"`, `"Person (sitting)"`, `"Cyclist"`, `"Tram"`, `"Misc"` (e.g., Trailers, Segways)

This dataset constitutes the benchmark for most of the papers and studies for autonomous driving perception tasks.

Chapter 2

State of the art

2.1 Introduction

The field of 3D object detection from point cloud is a quite new research field so there is not yet a method that has take over the other. In this chapter it is introduce a brief survive of the major research branches and the basis where they pose on.

Point clouds are a group of points where each point is composed of one coordinate in 3D space and a possible feature that can indicate the reflectance or other characteristic of that point. The most common way to generate a point cloud is using lidar or stereo cameras.

Lidar is a sensor that irradiates an environment with a rotating beam of laser rays and each ray when hit a surface and it is reflected it returns to the lidar and, measuring the time of flight, is possible to understand the position of the point. Repeating this many times and for all the points it is possible to obtain a group of the point that has more or less the shape of the environment.

The principle behind a stereo camera is the same as the human eye. First measuring the disparity of the position of an object in two parallel cameras then through a simple mathematical formula measures the depth of the object and then repeats this procedure to the entire scene.

The most difficult challenge in 3D object detection based on point cloud are:

- Irregularity: not all the surface have the same amount of point, some could be denser than other

- Unstructured: Point cloud data is not on a regular grid. Each point is sample independently so the distance between two adjacent points is not fixed. For example in an image that has a 2D fixed grid where the distance between a pixel and another neighborhood pixel is always fixed

- Unorderdness: The order of point in a point cloud are into stored in a specific order due to the fact that the intrinsic nature of this data makes useless to find it

The main family of NN that work on point cloud differentiates from each other regarding the approach to manage these three characteristics. They are structured grid-based, point-based, and mixed data types.

The structured grid-based NN tries to transpose and adapt the knowledge of the convolutional neural network in a tridimensional world. To achieve this goal two main approaches are adopted: Voxel-based network and multi-view network.

The voxel-based network takes the entire point cloud and then fill a 3D grid with the point and then apply 3D convolution pooling and a fully connected layer. The drawbacks of Voxel-based method is the high memory consumption due to the fact that the necessity to Voxilize the whole scene, the waste of resource because you need to convolve over empty Voxels and the lack of resolution because you have to find a trade-off between resolution and resources.

Multi-view based network the object is projected into many 2D planes and then it is applied a 3D image detection. The problem of that method is also a waste of resources and the loss of 3D depth while projecting the image. The advantages are: have more details and use of a solid method of detection like 2D detection.

The other family of detectors is the detection base directly on the point cloud. In this family, the main network is PointNet that is used as backbone

for many other networks. Due to the fact that points are unordered, PointNet is composed of symmetric functions. Symmetric functions are functions whose output is the same irrespective of the input order. PointNet is built on 2 basic symmetric functions: multi-layer perception (MLP) with learnable parameters, and a max-pooling function. The MLPs are a linear layer that works with the same parameters on each point creating a feature. The max-pooling layer is needed to aggregate the global feature. These symmetric functions are needed to work on three different main operations: sampling, grouping, and mapping. Sampling is reducing the number of points taking "the most relevant" points. Grouping is the operation that groups the nearest point, and then a mapping function that tries to map the features to a specific object class.

Mixed data types use two kinds of data like 2D images and point clouds to extract the 3D position of the object.

2.2 Structured grid network

2.2.1 VoxelNet

VoxelNet[3] is a network for 3D object detection based on Voxels. Voxels are a grid in a 3D dimensional space that encodes the geometrical feature of a point cloud.

VoxelNet divides the point cloud into equally spaced 3D voxels, encodes each voxel via stacked VFE (voxel feature encoding) layers, and then 3D convolution further aggregates local voxel features, transforming the point cloud into a high-dimensional volumetric representation. Finally, a Region Proposal Network (RPN) consumes the volumetric representation and yields the detection result. This efficient algorithm benefits both from the sparse point structure and efficient parallel processing on the voxel grid.

Voxel feature encoding (VFE) layer, which enables inter-point interaction within a voxel, by combining point-wise features with a locally aggregated feature. Stacking multiple VFE layers allows learning complex features for

characterizing local 3D shape information.

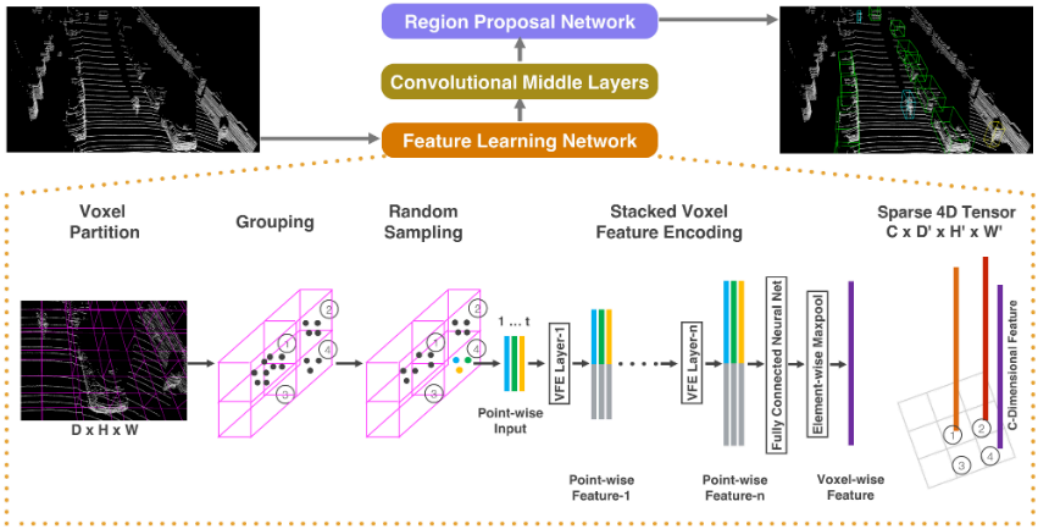


Figure 2.1: Structure of VoxelNet

VoxelNet is composed from three functional blocks:

1. Feature learning network,
2. Convolutional middle layers
3. Region proposal network

Feature learning network

The most complex functional block is the Feature learning network. It is subdivided into many sections:

- Voxel Partition: in this section, the 3D space is sub-divided into equally spaced voxels composed by voxel of equal dimension
- Grouping: group all the point inside each voxel
- Random Sampling: random sampling a fixed number of points in voxel that has a number of points higher than a certain threshold. This is done in order to reduce computation and increase the balance between the number of points in each voxel

- Stacked Voxel Feature Encoding: this section is the core innovation of the network. For each voxel first, it has been computed the local mean centroid of all the points. Then for each point inside the voxel, the relative offset from the centroid is evaluated obtaining an input feature for a fully connected network (FCN). The FCN is composed of a linear layer, a batch normalization (BN) layer, and a rectified linear unit (ReLU) layer. Then the points are max-pooled and the output of max-pooling is concatenated with the output of Relu layer

- Sparse Tensor Representation: group the non-empty voxel in order to obtain a light sparse 4D tensor, of size $C \times D' \times H' \times W'$ to reduces the memory usage and computation cost during backpropagation.

Convolutional Middle Layers

The convolutional middle layers aggregate voxel-wise features within a progressively expanding receptive field, adding more context to the shape description. It is composed of a 3D convolution, BN layer, and ReLU layer sequentially.

Region Proposal Network

The input to our RPN is the feature map provided by the convolutional middle layers. The network has three blocks of fully convolutional layers. The first layer of each block downsamples the feature map by half via convolution with a stride size of 2, followed by a sequence of convolutions of stride 1. After each convolution layer, BN and ReLU operations are applied. We then upsample the output of every block to a fixed size and concatenate to construct the high-resolution feature map. Finally, this feature map is mapped to the desired learning targets: (1) a probability score map and (2) a regression map.

Loss Function

The loss function is a weighted sum composed of the center location, dimension, and orientation of positive anchors and negative anchors.

Consideration

This network's approach poses the starting point for many networks in particular for the use of the Voxel Feature Encoding layer that has been replied to in different architectures.

2.2.2 Point Pillar

The goal of Point Pillar[4] is to make 3D object detection with only 2D convolutional layers. The name it's derived from the novel encoder that learns features on pillars (vertical columns) of the point cloud to predict 3D-oriented boxes for objects. For the authors the advantages of this approach are:

- there is no need to tune the binning of the vertical direction by hand (like for voxels);
- it is highly efficient because all key operations can be formulated as 2D convolutions which are extremely efficient to compute on a GPU;
- point pillar requires no hand-tuning to use different point cloud configuration so it can easily incorporate multiple lidar scans or even radar point clouds.
- It is faster comparing to the state of the art

The network is composed of three main stages:

1. A feature encoder network that converts a point cloud to a sparse pseudo-image;
2. A 2D convolutional backbone to process the pseudo-image into a high-level representation;

3. A detection head that detects and regresses 3D boxes.

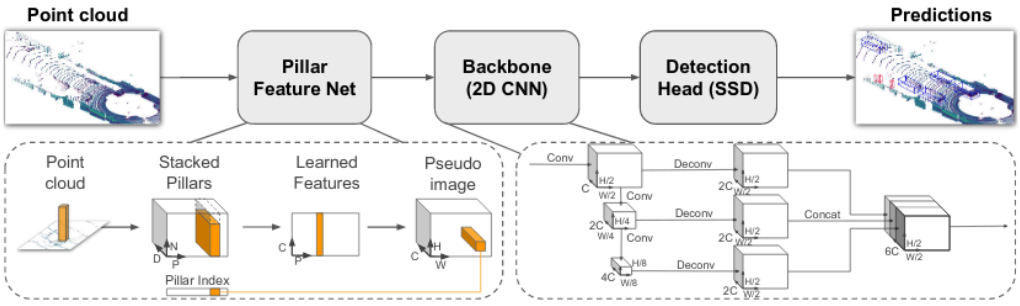


Figure 2.2: Structure of STD

Pointcloud to Pseudo-Image

In this section, the point cloud is converted to a pseudo-image. In order to do that these steps are applied:

The point cloud is discretized into an evenly spaced grid in the x-y plane, creating a set of pillars.

It has been evalutated the distance from the center of all the point inside the pillar and the distance from x, y center of the pillar so a point now have D=9 dimension $(x, y, z, x_c, y_c, y_c, x_p, y_p)$.

Inside each pillar, a linear layer, batch norm, and ReLu are applied (as Pointnet).

Backbone

In this phase, it is applied a backbone as VoxelNet compose of two sub-networks:

One top-down network that produces features at increasingly small spatial resolution;

A second network that performs upsampling and concatenation of the top-down features.

The top-down backbone is composed of a series of 2D convolutional layers and batch norm and ReLu that decrease each time the size of the output.

In the second part, all the features are upsampled using a transposed 2D convolution and then concatenate.

Detection Head

It has been used a Single Shot Detector (SSD).

Loss Function

The loss has been evaluated by computing a weighted sum of the loss of position, direction, and classification.

Consideration

The advantage of this network it that is light weighted, but this rise a lack of performance that it has to take into account.

2.2.3 STD: Sparse-to-Dense 3D Object Detector for Point Cloud

STD: Sparse-to-Dense 3D[5] is a two-stage 3D object detection framework. This network integrates advantages of both point-based and voxel-based and adds a 3D IoU prediction branch that increases the alignment between classification score and localization, achieving an important improvement. Their innovation is called the PointsPool layer. This layer is in charge of transforming the un-order points into a more compact feature. Another new element is the 3D IoU branch for predicting 3D IoU between predictions and ground-truth bounding boxes.

Proposal Generation Module (PGM)

Proposal Generation Module (PGM): the first step of PGM is the creation of spherical anchors. The choice of spherical anchor instead of cuboid anchors is derived by the consideration that a 3D object could be with any orientations. These spherical anchors have a fixed radius according to the class of the object. Then a 3D semantic segmentation network has involved to predict the class of

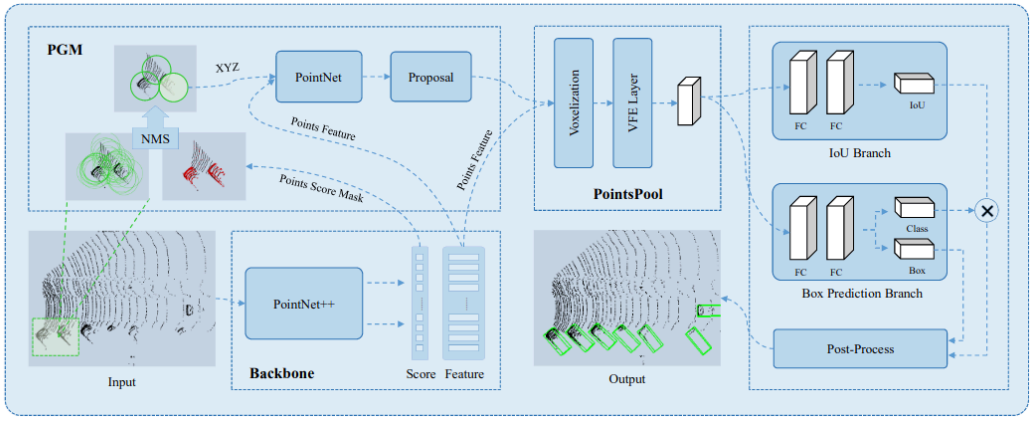


Figure 2.3: Structure of STD

each point and produce a semantic feature for each point. After that, a non-maximal suppression (NMS) is applied in order to remove redundant anchors. The final score of each anchor is the segmentation score on the center point. The IoU value is calculated based on the projection of each anchor to the BEV.

Proposal Generation Network: this section works on point in anchors (normalized by the anchor center coordinates), and semantic features like in PointNet after that an NMS based on classification score and oriented BEV IoU is applied to eliminate redundant proposals.

Assignment Strategy: For the IoU has been evaluated a new strategy called Points IoU is defined as the quotient between the number of points in the intersection area of both regions and the number of points in the union area of both regions.

Proposal Feature Generation

The goal of this section is to give semantic features from the segmentation network for each point and refined proposals, constitute compact features for each proposal.

In order to have a faster stage, it has been applied to a voxelization layer named PointsPool. PointsPool layer is composed of three steps.

In the first step, it has been randomly chosen N interior points for each

proposal with their canonical coordinates and semantic features as the initial feature. Then for each proposal, it has been obtained point canonical locations by subtracting the proposal center (X, Y, Z) values and rotating them to the proposal predicted orientation.

The second step is using the voxelization layer to sub-divide each proposal into equally spaced voxels. Then has been applied a voxel feature encoding like VoxelNet.

Box Prediction Network

The box prediction network has two branches for box estimation and IoU estimation.

Box Estimation Branch: In this branch, we use 2 FC layers with channels to extract features of each proposal. Then another 2 FC layers are applied for classification and regression respectively.

IoU Estimation Branch: first it has been applied to a 3D IoU. Then, each box's classification score is multiplied with its 3D IoU as a new sorting criterion.

Loss

The loss used is a multitask loss that is the sum between the loss of the proposal generation and the prediction loss. The proposal generation loss is the summation of 3D semantic segmentation loss and proposal prediction loss. The box prediction loss is almost the same as the proposal prediction loss with two extra losses, which are 3D IoU loss and corner loss.

Consideration

The interesting part of this network is the combining of a Voxel-based logic and a Point-based logic in order to achieve a greater improvement on the result.

2.2.4 SVGA-Net: Sparse Voxel-Graph Attention Network

SVGA[6] is a voxel-based network, so it divides the point cloud into some predefined space with a predefined shape. In this case, the point cloud has been divided into 3D spherical space with a fixed radius. The real innovation of this network is the use of an attention mechanism to extracted feature 3D. This is used in the voxel-graph network that first construct local and global graph for each voxel then it applies the attention mechanism providing a parameter supervision factor for the feature vector of each point. In this way, the local aggregated features are combined with the global point-wise features.
SVGA-Net architecture mainly consists of two modules: voxel-graph network and spare-to-dense regression.

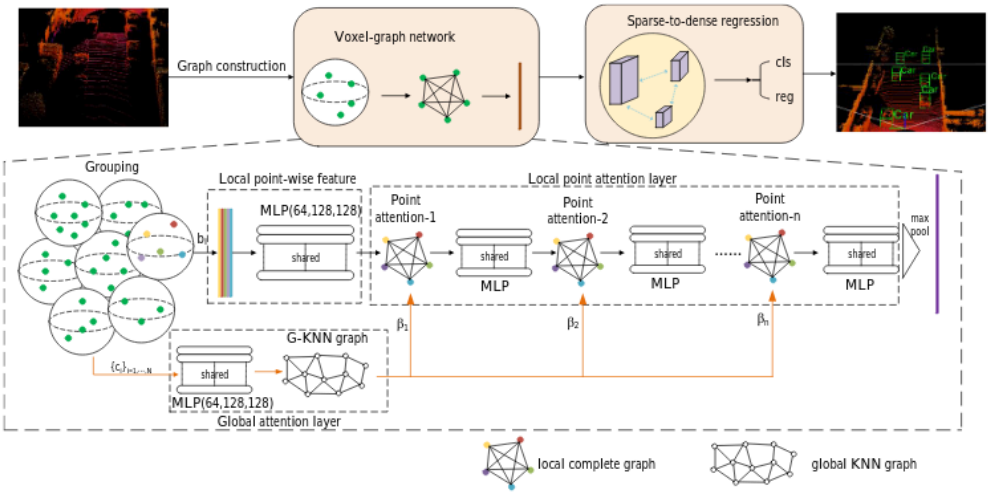


Figure 2.4: Structure of SVGA

Voxel-graph network

Spherical voxel grouping

The grouping phase has been done with farthest point sampling an iterative algorithm that searches the center of the sphere and then with it searches the neighborhood inside a fixed radius r. The Point cloud is now subdivided in N 3D spherical voxels $B = \{b_1, b_2, ..., b_N\}$.

Local point-wise feature

For each voxel with a MLP it has been extract the local point-wise features obtaining the local point-wise feature representation for each voxel sphere $F = \{f_i, i = 1, ..., t\}$.

Local point-attention layer

After that, the algorithm constructs a complete graph for each local node-set and a KNN graph for all the spherical voxels. The information on each node in aggregate according to the local and global attention score. The feature aggregation of $j - th$ node is represented as:

$$f'_j = \beta_m \cdot f_j + \sum_{k \in \sqcup(p_j)} \alpha_{j,k} \cdot f_{j,k}$$

f'_j denotes the dynamic updated feature of node p_j and f_j is the input feature of node p_j . $\sqcup(p_j)$ denotes the index of the other nodes inside the same sphere. $f_{j,k}$ denotes the feature of the $k - th$ nodes inside the same sphere. $\alpha_{j,k}$ is the local attention score between node p_j and the other nodes inside the same sphere. β_m is the global attention score from the global KNN graph in the $m - th$ iterations. $\alpha_{j,k}$ is evaluated as:

$$\alpha_{j,k} = softmax_j(f_j, f_{j,k}) = \frac{exp(f_j^T \cdot f_{j,k})}{\sum_{k \in \sqcup(p_j)} exp(f_j^T \cdot f_{j,k})}$$

Global attention layer

Its scope is to capture the global feature. First, it calculates the center of each voxel, and each center is learned by a 3-layer MLP to get the initial global feature. The KNN graph has been constructed for the N voxel sphere. For each node $f_{g,i}$, the attention score between node $f_{g,i}$ and its $l - th$ neighbor is calculated as follows:

$$\beta_m = \frac{f_{g,i}^T \cdot f_{g,i,l}}{\sum_{l \in \cup(f_{g,i})} f_{g,i}^T \cdot f_{g,i,l}}$$

where $\cup(f_{g,i})$ denotes the index of the neighbors of node $f_{g,i}$. m is the number of the point attention layers.

Voxel-graph features representation

After each local attention layer, a 2 MLP layer has been applied. Then in the end a max-pool is applied in order to aggregate the feature to obtain the final feature vector.

Sparse-to-dense regression

SDR module first applies three similar blocks to generate smaller spatial resolution from top to down. Each block consist of series of Conv2D layers, followed by Batch-Norm and a ReLU. The stride size of the Conv2D is set to 2 for the first layer of each block to down-sample the feature map by half, followed by a sequence of convolutions with stride 1. The output of this block is rename b_1, b_2, b_3. In order to combine high-resolution features with large receptive fields and low-resolution features with small receptive fields, the output of the second and third modules are concatenated with the output of the first and second modules after upsampling. Then a series of convolution operations with an upsampling layer are performed in parallel on three scale channels to generate three feature maps with the same scale size. The output of this block is rename F_1, F_2, F_3. After that the output b_1, b_2, b_3 is up sampled and combine with F_1, F_2, F_3 by element-wise addition. The final output F_s is obtained by concatenating the fused feature maps after a 3×3 convolution layer. F_s is taken as input to perform category classification and 3D bounding box regression.

Loss Function

The total loss is a multi-task loss composed of two parts, the classification loss L_{cls} and the bounding box regression loss L_{reg} balanced using a correction factor.

For the classification loss, it has been applied a classification binary cross-entropy loss.

For the regression loss, it has been applied a sum of the normalized distance between the point and the ground truth.

Consideration

This network use in a very interesting way the attention combining both local feature and global feature achieving a very good result.

2.3 Point based network

2.3.1 PointNet

PointNet[7] is a very important neural network because it represents a revolution in 3D object detection on point cloud. As said before a point cloud is a subset in a Euclidean space, so it has three main properties:

- Unordered (need operations that don't care about the order),
- Interaction among point (points have a geometric neighborhood)
- Invariance under transformations

In order to use these properties 3 different strategies have been adopted:

1. sorting input into a canonical order;
2. treating the input as a sequence to train an RNN, but augment the training data by all kinds of permutations;
3. using a simple symmetric function to aggregate the information from each point.

Asymmetric function Is a function where the output is invariant to the input order like, for example, sum and multiplication.

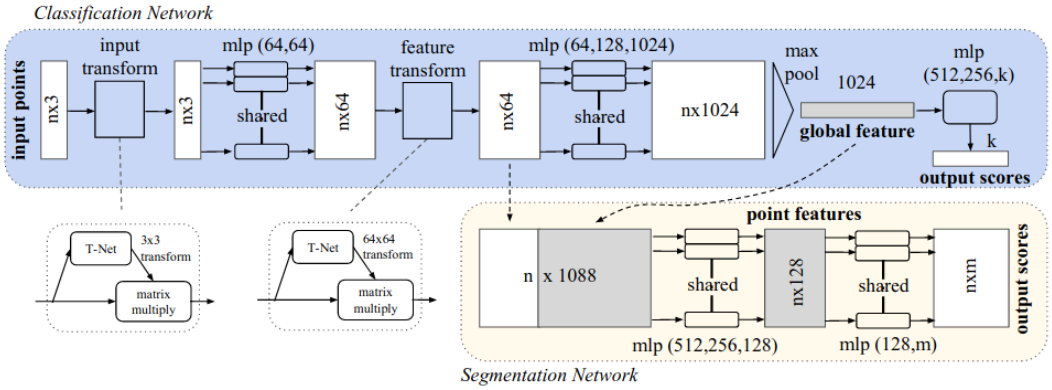


Figure 2.5: Structure of PointNet

Classification Network

The idea behind PointNet is to apply the same symmetric function to the entire set of points :

$$f(x_1, ..., x_n) = g(h(x_1), ..., h(x_n))$$

In particular h is a multi-layer perception network (MLP) and g a max pooling function. This constitutes the Classification Network whose goal is to classify the object of the point cloud.

After this phase, there is the Local and Global Information Aggregation. In this phase, the output from the above section ($[f_1, ..., f_K]$), (global feature) is concatenated with each of the point features (the output of a previous MLP).

Segmentation Network

Then, after sending this vector to other MLP and max-pooling layer, it extracts new per point features based on the combined local and global point features. The goal of this phase is to make semantic segmentation of the scene, so this phase is usually called Segmentation Network

Inside the Classification Network, there are two small networks called Joint Alignment Network. The goal of this network is to find a rotation matrix that has to rotate all the points or features in order to give a better orientation.

Consideration

This network works very well also if the number of points has been reduced but a problem introduced is that it does not try to group the point and the feature regarding the spatial distance between points. This has been overcome by PointNet ++.

2.3.2 PointNet ++

As said before PointNet is not able to take the local structures induced by the metric space points live in so it's limit the ability of the network to recognize fine-grained patterns and work in a complex scene.

The idea of PointNet ++ [8] is first work on each point as for PointNet and then aggregate the point feature capturing the local structure. In particular, the scene is partitioned into a set of points overlapping local regions and then the local feature is extracted capturing fine geometric structures; such local features are further grouped into larger units and processed to produce higher-level features. This process is repeated until it has been obtained the features of the whole point set.

The real challenge is how to create a group of neighborhoods and how to select the centroid of this group. In this network, the algorithm used for the selection of the centroids is the farthest point sampling (FPS) algorithm.

PointNet uses a multi-layer perception Network as a base concept in order to work for every single point.

$$f(x_1, ..., x_n) = \gamma\Big(MAX_{i=1,...,n}\{h(x_i)\}\Big)$$

f is invariant to input point perturbations and γ and h is the MLP.

The real innovation is that while PointNet uses a single max-pool layer to aggregate the feature. In this network, there are many sequential grouping of points and progressively abstract larger and larger local regions.

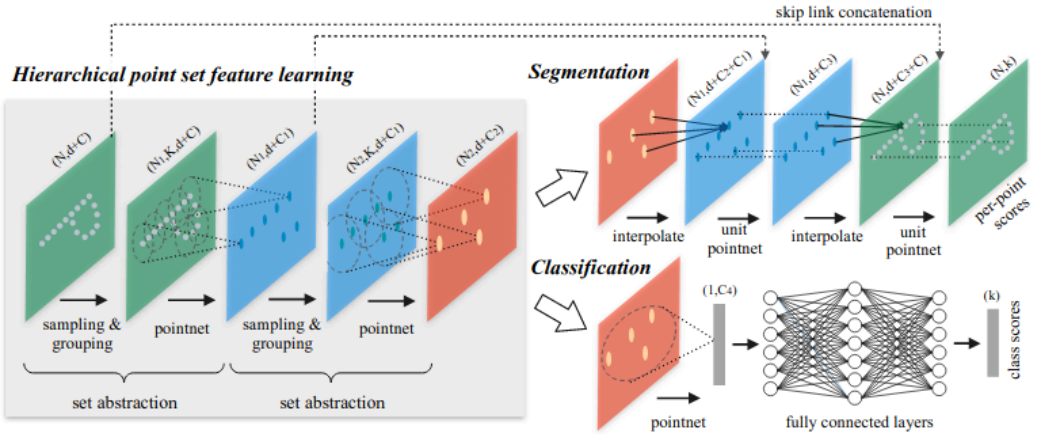


Figure 2.6: Structure of PointNet ++

Set abstraction level

The hierarchical structure is composed of a number of set abstraction levels. At each level, a set of points is processed and abstracted to produce a new set with fewer elements.

The single abstraction level is made of three key layers: Sampling layer, Grouping layer, and PointNet layer. It takes an $N \times (d + C)$ matrix as input that is from N points with $d - dim$ coordinates and $C - dim$ point feature. It outputs an $N' \times (d + C')$ matrix of N' sub-sampled points with $d - dim$ coordinates and new $C' - dim$ feature vectors summarizing local context.

The Sampling layer selects the centroids of local regions. To achieve this task ti use in a iterative way FPS to choose a subset of points $\{x_{i1}, x_{i2}, ..., x_{im}\}$ from $\{x_1, x_2, ..., x_n\}$ input point, such that x_{ij} is the most distant point (in metric distance) from the set $\{x_{i1}, x_{i2}, ..., x_{ij-1}\}$ with regard to the rest points. In this way the network is able to achieve better result because the centroids are generate not randomly but they depend from the data.

The grouping layer constructs local region sets by finding "neighboring" points around the centroids. It takes as input the entire point set of size $N \times (d+C)$ and the coordinates o fa set of centroids of size $N' \times d$. The output is groups of point sets of size $N' \times K \times (d + C)$, where each group corresponds to a local region and K is the number of points in the neighborhood of centroid

points. K is not a fixed value but it changes regarding the number of points in the surround of the centroid.

PointNet layer uses a mini-PointNet to encode local region patterns into feature vectors. In this layer, the input are N' local regions of points with data size $N' \times K \times (d + C)$. Each local region in the output is abstracted by its centroid and local feature that encodes the centroid's neighborhood. Output data size is $N' \times (d + C')$. The coordinates of points in a local region are firstly translated into a local frame relative to the centroid point: $\hat{x}(j)_i = \hat{x}(j)_i - \hat{x}(j)$ where $\hat{x}$ is the coordinate of the centroid. Then a simple PointNet is apply to classify the local region.

Grouping strategies

A challenge to develop this network is to find a Robust Feature Learning under Non-Uniform Sampling Density. The sampling has to work either where there are many points so and either where there is less point, in order to do that two different grouping strategies have been adopted:

Multi-scale grouping (MSG). A first way to capture multi-scale patterns is to apply grouping layers with different scales and then concatenate it

Multi-resolution grouping (MRG). The MSG approach above is computationally expensive since it runs local PointNet at large scale neighborhoods for every centroid point. In particular, since the number of centroid points is usually quite large at the lowest level, the time cost is significant. In order to reduce computational time in MRG. The features of a region at some level Li is generated by the concatenation of two vectors. One vector is obtained by summarizing the features at each sub-region from the lower level L_{i-1}. The other vector (right) is the feature that is obtained by directly processing all raw points in the local region using a single PointNet.

Point Feature Propagation for Set Segmentation

In a segmentation task, each point must be classified. During the many abstraction layers, the points are sub-sampled. In order not to lose the global information the feature are propagated. In order to propagate the features the feature point is interpolated with inverse distance weighted average based on k nearest neighbors. The interpolated features are then concatenated then are passed through a "unit PointNet", which is similar to one-by-one convolution in CNNs. A few shared fully connected and ReLU layers are applied to update each point's feature vector. The process is repeated until we have propagated features to the original set of points.

Consideration

This evolution of PointNet constitutes the real backbone for many networks for 3D object detection from the point cloud. In particular, the Set Abstraction Layer constitutes the real innovation of this network.

2.3.3 VoteNet

VoteNet[9] is an end to end network that the primary goal is object detection and segmentation. These networks use as backbone PointNet ++ and add the idea of Hough voting that are used to generate new points that lie close to objects centers, which can be grouped and aggregated to generate box proposals.

Hough voting is based on the concept of Hough transformation, an algorithm that translates the problem of detecting simple patterns in point samples to detecting peaks in parametric space. This concept is used to sample a set of seed points and generate votes from their features. These votes are designed to reach object centers in order to be easier aggregated through a learning module to generate box proposals.

VoteNet is composed of two parts: one that processes existing points to

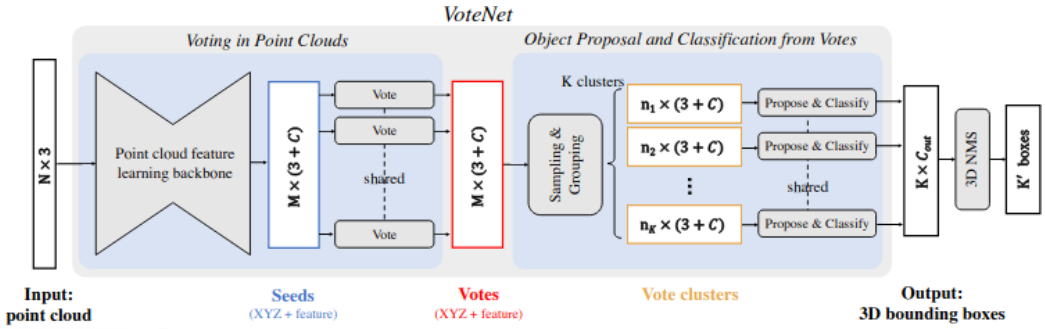


Figure 2.7: Structure of VoteNet

generate votes; and the other part that operates on votes to propose and classify objects.

Learning to Vote in Point Clouds

The goal of this piece of the network given as input a point cloud of size $N \times 3$, with a 3D coordinate for each of the N points, generate M votes, and each vote has both a 3D coordinate and a high dimensional feature vector. It is sub-divide into two steps:

Point cloud feature learning through a backbone network. The backbone uses in VoteNet is PointNet++. The output of this network is a subset of input points compose by M point and feature, $M \times (3 + C)$, where C is the number of features. Each seed point generates one vote.

Hough voting with deep networks from seed points. In traditional Hough voting, the votes (offsets from local key-points) are determined by look-ups in a pre-computed code-book. In this network, this feature has been adapted so the votes are generated with a deep network based voting module improve efficiency and accuracy. Specifically, the voting module is realized with a multi-layer perception (MLP) network with fully connected layers, ReLU, and batch normalization. The MLP takes seed feature and gives as outputs the Euclidean space offset $\Delta x_i \in R^3$ and a feature offset $\Delta f_i \in R^C$ from a centroid generate by the vote of the backbone. The predicted 3D offset Δx_i is explicitly supervised by a regression loss. The goal of this section is to translate the

feature and the point close to the centroid in order to have an easier recognition of the object.

Object Proposal and Classification from Votes

After the previous phase, the vote needs to be aggregate. To create the small cluster has been adopted a strategy of sampling and grouping according to spatial proximity using farthest point sampling-based the center of votes. In this way given K votes, it has been formed K cluster.

Proposal and classification from vote clusters

To aggregate the vote cluster it has been used a shared PointNet. First the vote locations has been transformed to a local normalized coordinate system by $z'_i = (z_i - z_j)/r$. Then an object proposal for this cluster $p(C)$ is generated by passing the set input through a PointNet like module:

$$p(C) = MLP_2\{max_{i=1,\dots,n}\Big\{MLP_1([z'_i; h_i])\}\Big\}$$

The votes from each cluster are independently processed by a MLP_1 before being max-pooled (channel-wise) to a single feature vector and passed to MLP_2 where information from different votes are further combined. We represent the proposal p as a multidimensional vector with an objectness score, bounding box parameters (center, heading, and scale parametrized) and semantic classification scores.

Loss function

The loss function is a multi-task loss that includes the voting loss, an objectness loss, a 3D bounding box estimation loss, and a semantic classification loss. The losses are weighted losses such that they are on similar scales.

The vote regression loss is an L1 distance. The objectness loss is a cross-entropy loss for two classes and the semantic classification loss is also a cross-

entropy loss of NC classes. The box loss is composed of center regression, heading estimation, and size estimation sub-losses.

Consideration

This network can be considered as the state of the art for 3D object detection, nevertheless, it has some problems in grouping layers in an outdoor environment. Further improvement and adjustment have been made in order to transpose this network in a sparse environment. A deeper description of this network has been provided in chapter 4.

2.3.4 3DSSD

VoteNet has been designed to work in a dense point environment like a room where the points are relatively near from each other, the goal of 3DSSD[10] is to adapt the VoteNet to work in a sparse environment like an urban environment. In order to do that some change has been adopted. The main change has been derived by observing that with the furthest point sampling based on 3D Euclidean distance (D-FPS) the foreground instances with few interior points may lose all points after sampling. Consequently, it is impossible for them to be detected In order to overcome this issue a novel sampling strategy based on feature distance, called F-FPS has been adopted. This sampling strategy has been merged with the D-FPS. With this method, it has been possible to consider not only spatial distance but also semantic information of each point during the sampling process.

$$C(A, B) = \lambda\, L_d(A, B) + L_f(A, B)$$

$L_d(A, B)$ and $L_f(A, B)$ represent L2 XYZ distance and L2 feature distance between two points and λ is the balance factor. This combination's called Fusion Sampling (FS) and has the advantages to retain more positive points for localization and keep enough negative points for classification as well.

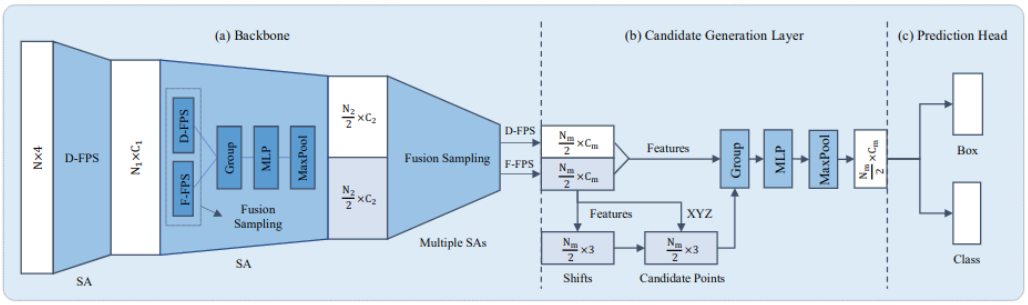


Figure 2.8: Structure of 3DSSD

Another innovation in this network is the Box Prediction Network that modifies the SA layer after the FS layer in order to make this process faster. In this network, it has been introduced the candidate generation layer (CG) before our prediction head, which is a variant of the SA layer. In this layer, it has been used only the points generate by the F-FPS. Then as VoteNet both the surrounding features and the surrounding candidate point are extract and an MLP is applied.

Anchor-free Regression Head

In the regression head, for each candidate point, it has been predicted the distance (dx, dy, dz) to its corresponding instance, as well as the size (dl, dw, dh) and orientation of its corresponding instance. Since there is no prior orientation of each point, it has been applied a hybrid of classification. An equally split orientation angle bins predefined and classify the proposal orientation angle into different bins.

3D Center-ness Assignment Strategy

For each candidate point, it has been defined as a Center-ness label through two steps. First, it has been determined whether it is within an instance l_{mask}, which is a binary value. Then has been drawn a Center-ness label according to its distance to 6 surfaces of its corresponding instance. The Center-ness label is calculated as

$$l_{ctrness} = \sqrt[3]{\frac{min(f,b)}{max(f,b)} \times \frac{min(l,r)}{max(l,r)} \times \frac{min(t,d)}{max(t,d)}}$$

where (f, b, l, r, t, d) represent the distance to front, back, left, right, top and bottom surfaces respectively. The final classification label is the multiplication of l_{mask} and $l_{ctrness}$.

Loss Function

The Loss Function is composed of weighted sum of classification loss, regression loss and shifting loss.

Consideration

The Loss Function is composed of a weighted sum of classification loss, regression loss, and shifting loss.

2.4 Mixed type network

2.4.1 Frustum PointNet

This method combines both 2D object detection from images and 3D object detection from point cloud. Frustum PointNet[11] exploit the advantages of 2D object detection to defines a 3D search space for the object.

exploit the advantages of 2D object detection to defines a 3D search space for the object. This network is composed of 3 stages:

1. Frustum proposal that extracts the 3D bounding frustum of an object by extruding 2D bounding boxes from image detectors;

2. A 3D instance segmentation that applies PointNet;

3. Amodal 3D network that predicts the 3D mask of the object of interest and regression network estimates the amodal 3D bounding box.

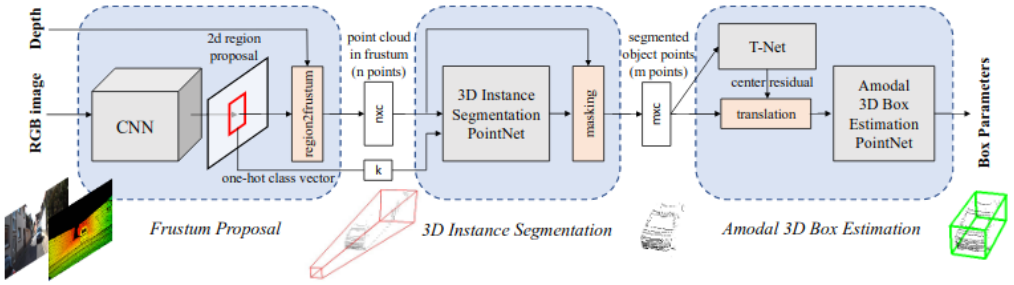


Figure 2.9: Structure of Frustum PointNet

Frustum Proposal

First with a 2D object detector propose 2D object regions in RGB images, then knowing the camera matrix is possible to lift the 2d bounding box and create a frustum and fill it with the point in the point cloud. Due to the fact that the generate frustums have different directions they are rotating them toward a center view such that the center axis of the frustum is orthogonal to the image plane in order to improve the rotation invariance of the algorithm.

3D Instance Segmentation

The network takes a point cloud in frustum and predicts a probability score for each point that indicates how likely the point belongs to the object of interest. Note that each frustum contains exactly one object of interest. This part of the network is based on PointNet. This network also uses the information given by the 2D detector concatenating it to the intermediate point cloud features. After 3D instance segmentation, points that are classified as the object of interest are extracted. Then the coordinates of the extracted point are normalized. The point cloud is then transformed into a local coordinate by subtracting XYZ values by its centroid.

Amodal 3D Box Estimation

Given the segmented object points (in 3D mask coordinate), this module estimates the object's oriented 3D bounding box and it is composed of a

Learning-based 3D Alignment and an Amodal 3D Box Estimation.
Learning-based 3D Alignment: To estimate the true center of the complete object is applied a T-Net derived by PointNet and then transform the coordinate such that the predicted center becomes the origin.
Amodal 3D Box Estimation: This is similar to the classification network but it predicts the 3D box. The center residual predicted by the box estimation network is combined with the previous center residual from the T-Net and the masked points centroid to recover an absolute center.

$$C_{pred} = C_{mask} + \Delta C_{t-net} + \Delta C_{box-net}$$

The angle and the size is evaluated evaluating the score of predefined bins.

Loss Function

In order to have an optimization of the the three nets involved (3D instance segmentation PointNet, T-Net and amodal box estimation PointNet) it has been used a multi-task losses.

$$L_{multi-task} = L_{seg} + \lambda(L_{c1-reg} + L_{c2-reg} + L_{h-cls} + L_{h-reg} + L_{s-cls} + L_{s-reg} + \gamma L_{corner})$$

L_{c1-reg} is for T-Net and L_{c2-reg} is for center regression of box estimation net. L_{h-cls} and L_{h-reg} are losses for heading angle prediction while L_{s-cls} and L_{s-reg} are for box size. Softmax is used for all classification tasks and smooth L1 loss is used for all regression cases.

Consideration

In order to have an optimization of the three nets involved (3D instance segmentation PointNet, T-Net, and amodal box estimation PointNet) it has been used multi-task losses.

Chapter 3

Stixel

Perception is a very important task for the self-driving car because without understanding the environment is impossible to move through. So, to have a better and more reliable view of the surrounding areas there are use many sensors like cameras, stereo cameras, and LIDARS. In order to ensure a complete view, these sensors are installed in many places of the cars. They allow to improve accuracy and redundancy, but they cause an increase in data that has to be transmitted. An autonomous vehicle has to have a fast as possible response to the stimulus provided by the sensors, but if the amount of data is too big it is very hard to ensure real-time computing. So the amount of data transmitted has to be reduced but without any loss of information. To overcome this has been created the stixel a medium-level representation that overcomes the gap between the pixel and the object level retaining the underlying information at the same time.

To attempt all the task that an autonomous vehicle has to do, the data should be:

1. compact: offering a significant reduction of the data volume;
2. complete: information of interest is preserved;
3. stable: small changes of the underlying data must not cause rapid changes within the representation;

4. robust: outliers must have minimal or no impact on the resulting representation.

Stixel has been thought especially for self-driving cars to work in a road environment. The geometry in human environments is dominated by two basic types: Horizontal and vertical planar surfaces. Horizontal surfaces generally correspond to the ground, i.e. roads, sidewalks, or soil, the vertical ones relate to objects, such as solid infrastructure, pedestrians, or cars. The most relevant that has been detected are vertical surfaces because thanks to the knowledge of vertical surfaces it has been possible to achieve many tasks for autonomous driving.

A stixel can is a small rectangle that starts from the base of the road and then rises until the top of the obstacle summarizing the vertical surface of an object.

It is a medium level representation that allows structured access to the scene data independent of the particular application without neither being too specific nor too generalizing. Stixels provides compressed and structured access to all relevant visual content of the scene. this type of compressed data can be used for a multitude of automotive vision applications, including object detection, tracking, segmentation, localization, and mapping.

In this thesis, the stixels are used for 3D object detection. In the following paragraphs, it explains the procedure of creating starting from a point cloud generated by the LIDAR sensor or from a depth map generate from a stereo or mono camera.

Even if point cloud and depth map have different characteristics the only thing that the procedure proposes a change from one type of data to another is the plane fitting. This is done in order to exploit the advantages of this type of data and to overcome the weakens.

3.1 State of the Art

The use of the stixel as a medium level representation is a filed that is not so much explored even if the first idea of stixel has been presented in 2009 by Hernán Badino [12]. In this paper, it has been defined the first idea of the stixel and the requirement for a medium level data representation.

In the initial approach presented in 2009, the Stixel World is constructed by cascading multiple independent algorithms: mapping disparities to occupancy grids, a free space computation, a height segmentation, and a final Stixel extraction step. However, such cascade is prone to errors, e.g. missed objects in the free space computation can not be corrected in subsequent steps. Further, the proposed scheme contains multiple thresholds and non-linearities. The major limitation is that this algorithm takes into account only the first obstacle along every viewing angle can cause missing relevant objects. the major improvement proposes by Pfeiffer [13] is the creation of an algorithm that allows multiple Stixels along every column of the image. Another improvement is the adding of information related to each stixel, like for example the class of the object of the stixel or further metadata. Pfeiffer applies the concept of stixel not only depth math extract by a camera but also on point cloud for LIDAR.

In 2012 Rodrigo Benenson [14] propose a fast method for generating stixel to detect pedestrian without depth map. The assumption in this method was that the object height is known and class-dependent. So identifying the object and assuming know the height of the object is possible to identify how far it is. The problem with this method is that it recognizes only one object and if there is some occlusion it does not perform well.

In the *Semantic Stixels: Depth is Not Enough* [15] [16] proposed by Lukas Schneider and Marius Cordts the concept of stixel proposed by Stixel world is fused with the information provided by a segmentation neural network in order to obtain a compact 3D information of the environment.

The previous works have been improved in *Slanted Stixels: Representing San Francisco's Steepest Streets* [17] taking into account non-flat roads and slanted objects and achieving real-time computation capabilities with only a slight drop inaccuracy.

From 2018 a new way to generate stixel has been started. it is base on the concept of fusing LIDAR and camera data improving both the geometric and semantic accuracy and reducing the computational overhead. In *Improved Semantic Stixels via Multimodal Sensor Fusion*[18] [19] the concept of Stixel is transposed into the LIDAR domain to develop a compact and robust mid-level representation for 3D point clouds.

The method purpose in this thesis is focused on the stixel generate by depth map and point cloud. The stixel contains only the depth information without the adding of object class and other information due to the fact that the goal of this stixel generation algorithm is to generate stixel that has to be feed into a neural network based on 3D object detection from a point cloud.

3.2 Procedure of creation

The algorithm proposed in this thesis is articulated in three phases:

1. Plane fitting: fit the plane in order to remove the point that lies in the ground;

2. Matrix depth creation: the creation of a matrix that for each cell contains a depth value;

3. Stixel creation: the creation of the stixels.

The algorithm has been tested both on point cloud generated by lidar and both on depth map generated by Neural Network starting from stereo or mono images. The only change between the procedure on point cloud and depth map is the procedure of plane fitting. This is due to the fact that the Point cloud

generates by lidar are more sparse than the depth map generated by stereo cameras.

In lidar point cloud the procedure to delete the plane is driven by the Ransac algorithm, this algorithm tries to fit the best plane discarding the outliers and it works well when the number of points is relatively small due to the high computational cost. Depth map generates by the stereo camera are denser due to the fact that the algorithm associate at each pixel a depth so in an image with size [375, 1242] we obtain 465 750 depth points instead of 40 000 points generates from a lidar, this means a huge increase of computational size. In order to exploit this characteristic a different kind of algorithm has been used.

3.2.1 Plane fitting

Plane fitting is the first step in the Stixel creation. This step is relevant because the stixel ideally should represent only the shape of the vertical object so in order to achieve this result the ground must be deleted otherwise a huge number of small stixel is created, and this is not a good thing because the goal of the stixel is to contain only the interest object and not the ground.

As explained in the section before, there is presented two way of deleting the ground regarding the type of data as input. This differentiation has been made in order to exploit at the best the qualities of these two data formats and improve the result.

Plane fitting on lidar point cloud

To fit the plane for lidar point it has been used an algorithm call Ransac. This algorithm iterative search for the best equation of the plane that fit the most number of points.

The first step of the algorithm is to select the sub-sample of the point where Ransac has to find the plane. It is useless to apply the Ransac search to the whole set points because due to the research task it can be right to assume that the points above a certain threshold can be declared not part of the ground.

This threshold cannot be a fixed value because the algorithm has to take into account the possibility of change of the slope of the road. So the threshold to take the first set of points has to increase according to how far the point is from the car, farthest is the point higher is the level of acceptance.

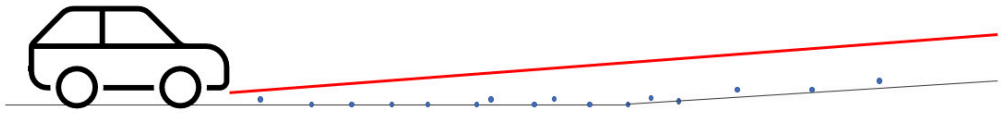


Figure 3.1: Representation of threshold on selecting points on ground

So after a subsample on the first set of points, Ransac could be implemented. It is composed of a for loop that iterative do this kind of operation:

1. from the subset of point select 3 random points;
2. Given these 3 points evaluate the equation of the plane that passes through these points through SVD factorization;
3. counts the number of remaining points which Euclidean distance from the plane previously evaluated is less than a certain threshold.

After a fixed number of iteration take the three points that fit the plane that have a higher number of points that Euclidean distance is less than a certain threshold. So after evaluating these three points find again all the set of points that Euclidean distance is less than a certain threshold and apply for the last time the Singular Value Decomposition to find the best plan that fits all the points.

The last procedure is to delete all the points that Euclidean distance is less than a certain threshold from the last plane finding.

After a few experiments, it comes out that a single plane fitting does not delete all the ground points because the lidar has a range between 50/60 meters, and in a ray of that distance the road can have several changes of slope. So in

order to adapt the algorithm to the various change of road, it has been created a grid that subdivides the surface into squares with equal dimensions and for each square Ransac algorithm has been evaluated running of the sub-set of points that projection on x-y plane lie inside the square.

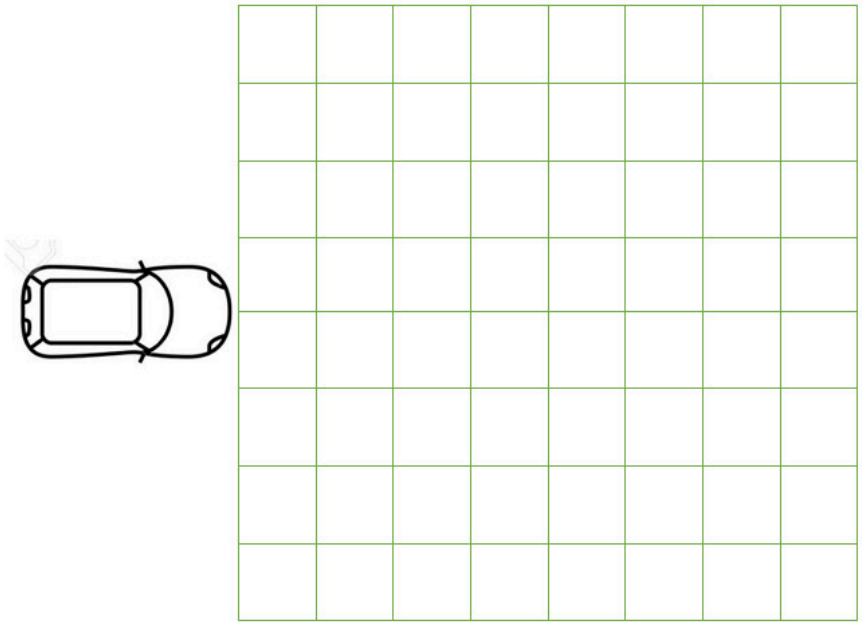


Figure 3.2: Representation of segmentation of the piano

(a) points projected on cam 2 (b) ground points

Figure 3.3: LIDAR points

Plane fitting on stereo image

In order to delete the plane from depth evaluating on stereo images a different kind of algorithm has been used. This is due to the fact that the Ransac algorithm evaluated in many points consumes too many resources and because thanks to the fact that a depth map is a dense map another more convenient approach can be used.

Figure 3.4: Obtained points on vertical surface

In order to extract the depth map from the stereo camera, it has been used the network described in *Pyramid Stereo Matching Network*[20]. For the depth map from a mono camera, the network applies to image of Kitti cam 2 is *From Big to Small: Multi-Scale Local Planar Guidance for Monocular Depth Estimation* [21].

The algorithm used in this thesis is called Vertically Local Disparity Histogram (VLDH). The goal of VLDH is to create a binary mask that extracts the pixel that has a disparity that is related only to the object excluding the road surface and the surrounding. The working principle is that if a set of depth in the pixel in the same column have more or less the same disparity this group of pixel belongs to an object. Otherwise, if the greater number of the pixel has a disparity that changes a lot from the disparity of the first pixel in the set it means that group of pixels does not belong to an object but it could be a road or background.

The first thing to do is to create a mask of zeros with dimensions of the depth map. This algorithm subdivides each depth images into columns of fixed size (ex 1 pixel) and for each column apply this procedure starting from the bottom line of pixels :

1. extract the depth for the pixel j;

2. count the number of pixels in the next N pixel that depth is in a range of $\pm\Delta d$ of the depth on pixel j;

3. If this number is greater that a threshold k :

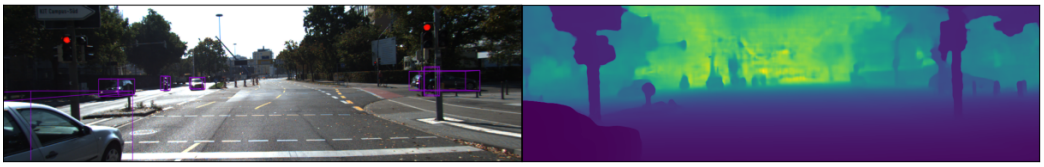
   - set at 1 all N points in the mask and jump at the point $j + N$ restarting the algorithm;

   If this number is smaller that a threshold k :

   - jump at the point $j + 1$ and restart.

After this procedure in order to extract the points of interest in the depth map the only operation to do is just multiply the mask with the depth map and extract the points that have a depth that is different from zero.

In order to have a better parallelization and improve the execution of the algorithm, it is possible to work in the entire raw creating a sub-mask that memorizes the points where the algorithm has to work or the points that are already set to one. In python, this improves the computation time from 5 seconds for an image to 0.06 seconds.



(a) original cam 2 image (b) depth map

Figure 3.5: Starting images to generate mask

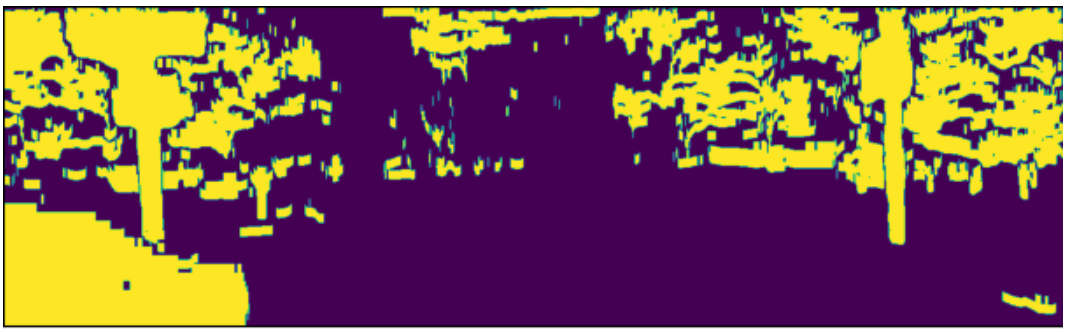


Figure 3.6: Obtained mask with VLDH

3.2.2 Matrix generation

The second step of the creation of the stixel in this procedure is the creation of a discretization matrix. The stixels are created working on the points that are projected on the image generated by cam 2. A single stixel has a fixed resolution with height 8 pixels and width 4 pixels. This means that a stixel can be thought like a composition of small bricks.

Figure 3.7: Representation of grid of stixel

In order to create these small bricks, the entire picture is divide into $N \times M$ cells and these cells are filled with the projected point cloud projected points. Due to the sparsity of the point cloud and due to the fixed resolution of the matrix many scenarios can be:

- fill with points of same object
- fill with points of different objects
- empty

Then for each cell, it has to be selected a depth that has to summarize all the depth of all the points inside. For this task different ways have been tried.

Average matrix

The first simple way to summarize the distance of the points inside the cell is to make the average of the depth. This simple method has a fast implementation but it has not enough discretization. If the points inside a cell have a belong

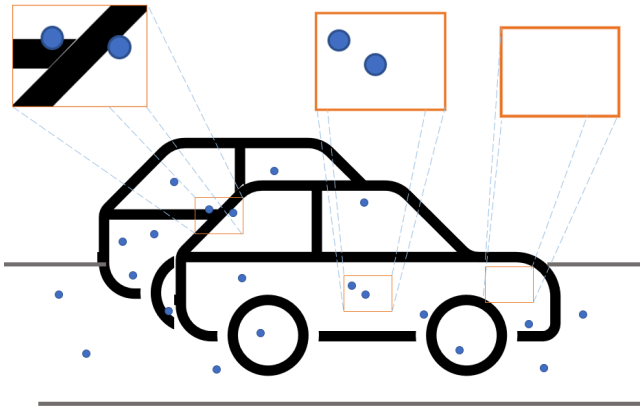


Figure 3.8: Representation of cells scenario

to different objects that have different distances, this algorithm assigns at this cell a distance that is in the middle between the two objects, creating a small stixel that is not associated either with the object and background.

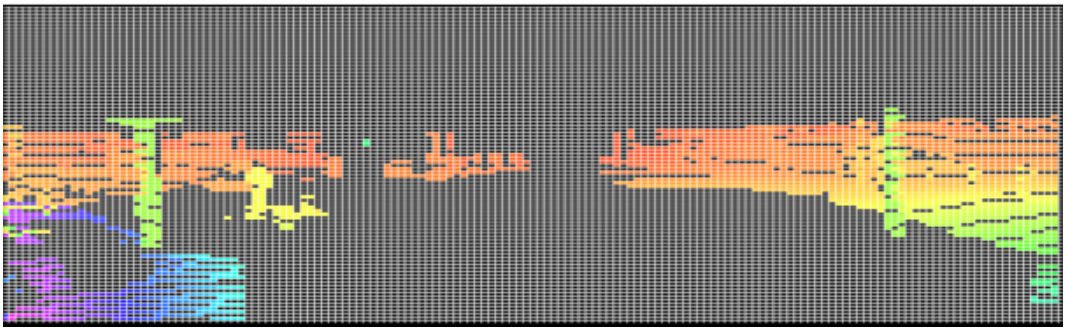


Figure 3.9: Average matrix results

Histogram matrix

In order to overcome the issue of the average matrix a new way to develop the base matrix has been evaluated and this is has been called the Histogram matrix. In the histogram matrix for each cell, it has assigned a value using the average of the depth of the most relevant cluster of the depth of points inside each cell.

Given a set of points that are inside a cell i, j: $S_{[i,j]}$

The algorithm used in each cell can be summarized as:

```
if S[i,j] = ∅ =>
    M[i,j]= 0
if |max(S[i,j])-min(S[i,j])| < d_threshold =>
    M[i,j] = mean(S[i,j])
else:
    if number of items of S[i,j] = 2 =>
        M[i,j] = min(S[i,j])
    else:
        S[i,j]> = {x ∈ S[i,j] | x > mean(S[i,j])}
        S[i,j]< = {x ∈ S[i,j] | x < mean(S[i,j])}
        if (number of items S[i,j]> ) > (number of items S[i,j]<) =>
          M[i,j] = mean(S[i,j]>)
        if (number of items S[i,j]> ) < (number of items S[i,j]<) =>
          M[i,j]= mean(S[i,j]< )
        if (number of items S[i,j]>) = (number of items S[i,j]<) =>
            if |max(S[i,j]>)-min(S[i,j]>)| > |max(S[i,j]<)-min(S[i,j]<)| =>
              M[i,j] = mean(S[i,j]<)
            if |max(S[i,j]>)-min(S[i,j]>)| < |max(S[i,j]<)-min(S[i,j]<)| =>
              M[i,j]= mean(S[i,j]>)
```

This simple flow chart allows assigning to each cell of the matrix the most suitable depth.

3.2.3 Stixel creation procedure

After the creation of the depth matrix, the last step of the algorithm is the creation of stixels. The creation of stixel is the merging in one stixel of all the

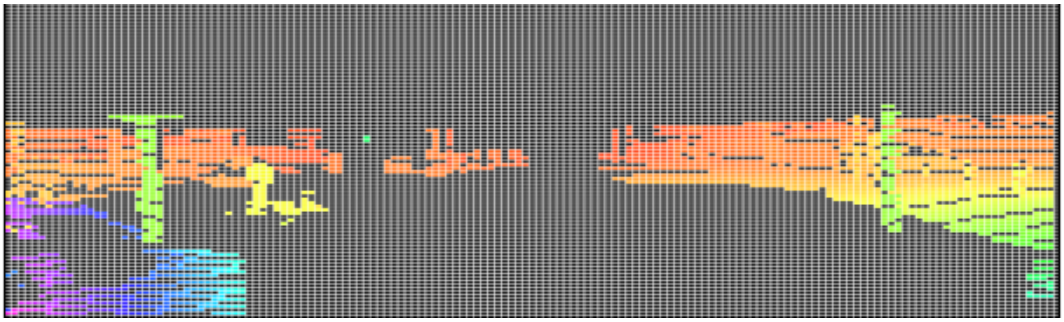


Figure 3.10: Histogram matrix results

cells of the matrix in the same column that has a depth that is more or less similar.

The algorithm of stixel creation can be summarized as a state machine with two states:

- new = 1: allow the creation of new stixel ;
- new = 0: allow to keep the previous stixel and to merge cells of the matrix.

This machine has also three working phases:

- create stixel where a new stixel is created;
- update stixel: where the height of the stixel is increase;
- update empty count where the counter of the number of empty cells is updated in order to set a maximum number of consecutive empty cells.

The procedure starts in $new = 1$ state and the condition that affect the transition from one state to another is the shift of the cells inside the matrix M driving by the for cycle.

Starting from $M[0,0]$ and $new = 1$ the state machine is ready to create a new stixel and when the cycle for moving toward the column find a cell where $M_{[i,j]} > 0$ the algorithm start to create a new stixel and it goes to state $new = 0$.

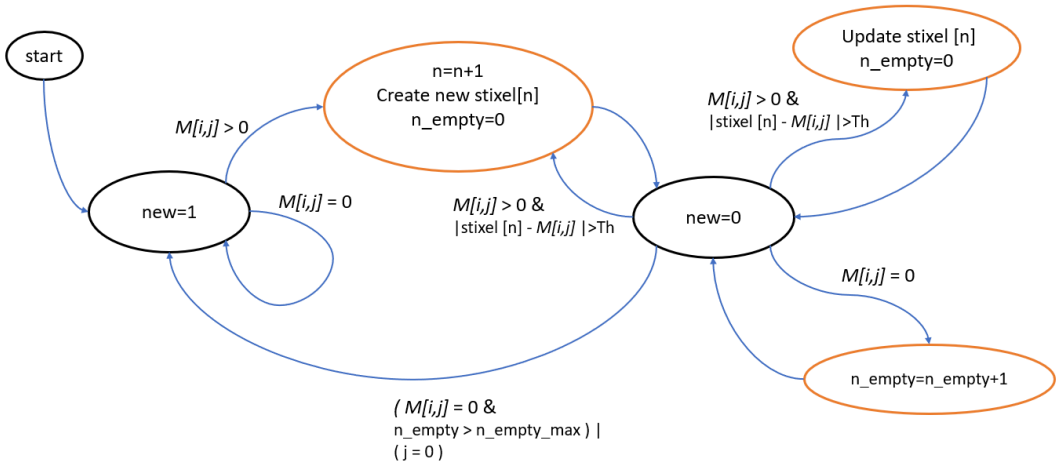


Figure 3.11: Algorithm of stixel creation

Starting from $M[0,0]$ and $new = 1$ the state machine is ready to create a new stixel and when the cycle for moving toward the column find a cell where $M_{[i,j]} > 0$ the algorithm start to create a new stixel and it goes to state $new = 0$.

Here there is several options:

- $M_{[i,j]} > 0$ and it is close to the depth of the stixel so the cell can be join to the stixel;
- $M_{[i,j]} > 0$ and it is far to the depth of the stixel so the cell is attach to a new stixel;
- $M_{[i,j]} = 0$ so it increase the counter of the empty cells.

The counter of the empty cells has been designed in order to overcome the sparsity of the point cloud that sometimes produces an empty cell inside a matrix M. In fact, there is the possibility that in a column of stixel, even if they below the same object, there could be one o more empty cells in the middle. In order to have a better aggregation, it has been chosen to count the number of empty consecutive cells, and if this number is above a certain threshold it starts the creation of a new stixel.

The counter of empty cells has to be reset when it starts the creation of a new stixel or when it updates a previous stixel.

The operation of updating the stixel means, after check if the difference of a new value and the value of the stixel is under a certain threshold, the update of the height of the stixel and update the value of depth of the stixel making an average between the value of the depth of stixel and the value of $M_{[i,j]}$.

The passage from $new = 0$ to $new = 1$ is determined if the number of empty cells in a column is above a certain threshold or if the algorithm starts a new column.

After scanning all the cells the algorithm obtains an array of stixels which is possible to save in many ways:

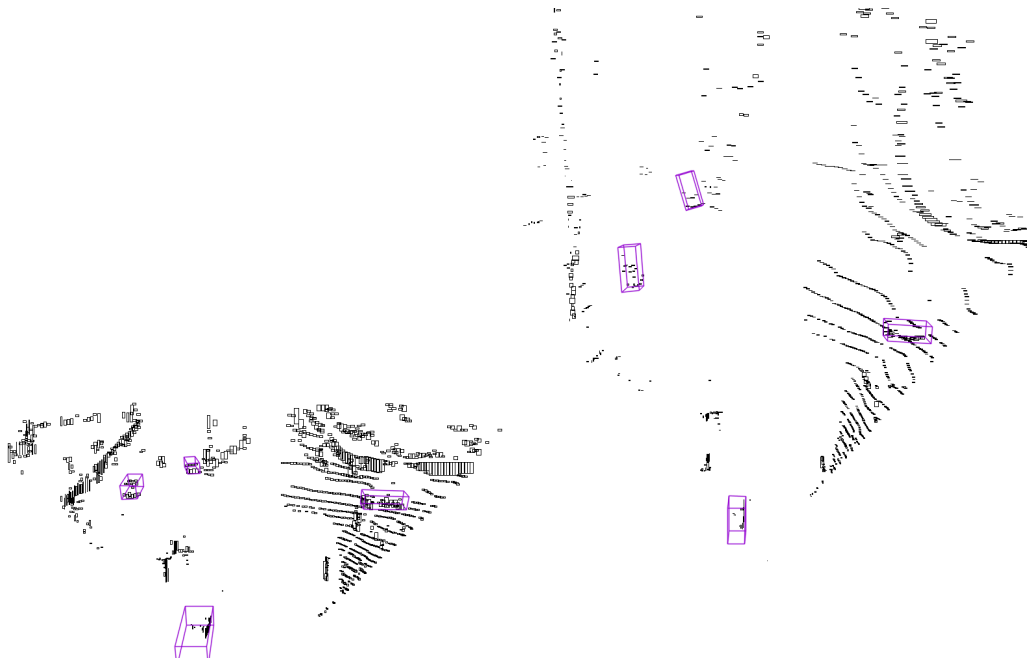
- As four points that compose the vertices of stixels;
- As two points that compose the opposite vertices of stixels;
- As one point of the center of the stixel and the height and width of this stixel.

3.3 Results

The stixel summarizes in a good way the depth of the map and the point cloud. And also in 3D and in bird's eye view, it is possible to notice that the result is quite confident to identify the position of the relevant objects.

Figure 3.12: Stixel result

An important analysis is the distribution of the number of stixel per image. First, it has been examined the stixel coming out from the algorithm apply to the point cloud.



(a) stixel in 3D prospective (b) stixel in BEV

Figure 3.13: Stixel in 3D prospective and stixel in BEV

In this analysis it come out that the number of stixel per image has:

- average: 1134;
- standard deviation: 367;

This small standard deviation implies that the number of stixel per image is quite concentrated. One problem is the outliers. The images that have a low number of stixel is due to the fact that in some scene the car runs in the highway without founding any cars so the only stixel that are present are the one come from guard rail so they are not a big number. More important is the analysis of the images that have a high number of stixel. This high number is due to the scatter generate in a scene where the car is crossing a forest so the irregular form of the trees produces a high scatter and this is capture by the stixels due to the fact that they are all vertical objects.

Although despite this outliers the number of stixel is around 1100 and the number of point on the same images is around 40 000 so this cause a reduction of 20 times fewer points.

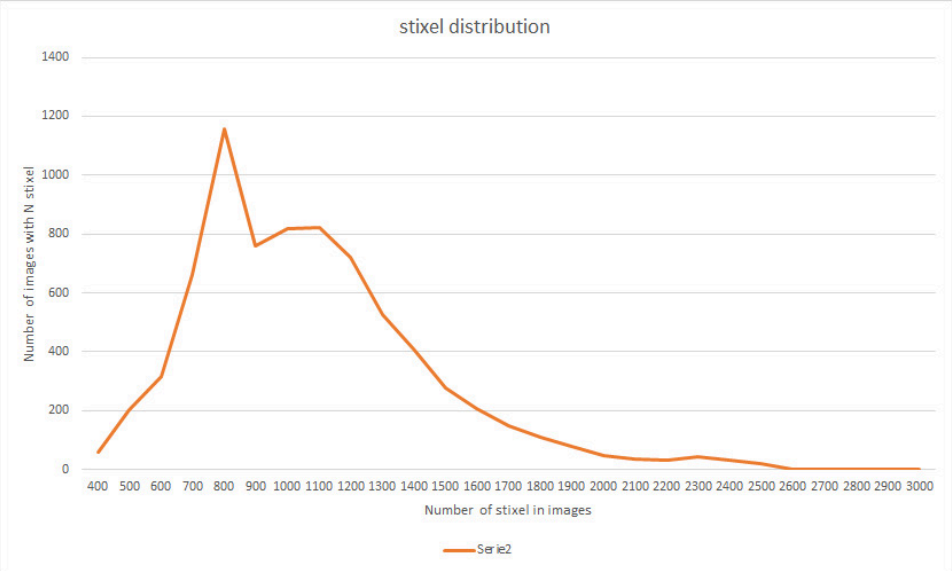


Figure 3.14: Distribution of stixel generate from lidar point cloud

Watching the number stixel generate starting from stereo depth map it is possible to see that have:

- average: 580;
- standard deviation: 105;

This distribution is more close due to the fact that the algorithm VLDH, for the elimination of the ground, also delete the background, where it changes to much so the number of stixel, is also reduced because only with this filtering only the points that are more or less at the same distance keep, so, thanks for that it is possible to generate less, but bigger stixel than the one generated from a point cloud.

This is confirmed also watching the distribution of the stixel generate starting from the depth map generate by mono camera that have:

- average: 580;
- standard deviation: 105;

This is more or less the same distribution of the stixel generate from the stereo depth map.

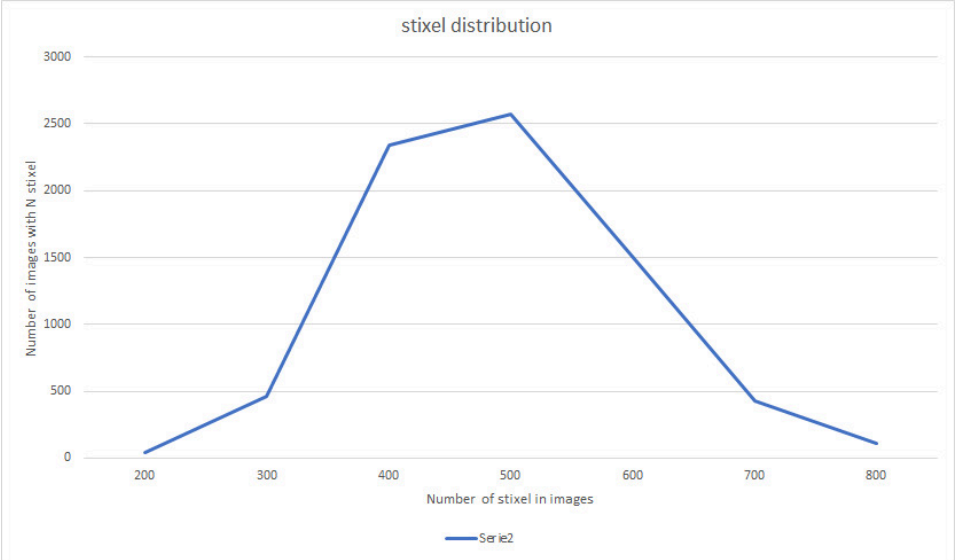


Figure 3.15: Distribution of stixel generate from stereo depth map

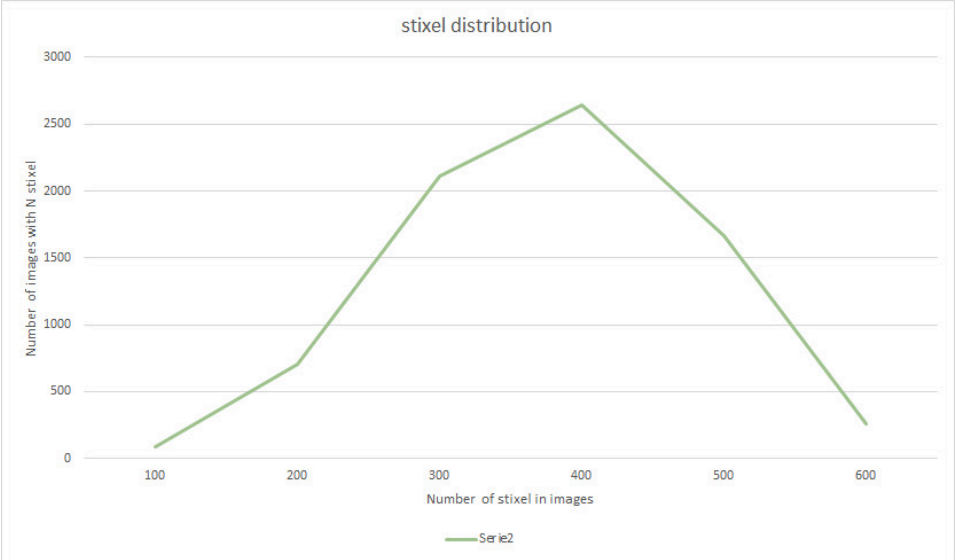


Figure 3.16: Distribution of stixel generate from mono depth map

So, it is possible to say that the stixel reduced a lot of dimension of the information and now it is important to see if this reduction implies a reduction of performance in the 3d object detection.

Chapter 4

3D StixelNet

In this chapter, it is present the Neural network use for 3D object detection. This network is an adaptation of VoteNet, modified in order to be more suitable for spare point cloud, like the one generated from LIDAR in the outdoor environment.

The innovation is the adaptation of a neural network that works on Point Cloud to working on stixels.

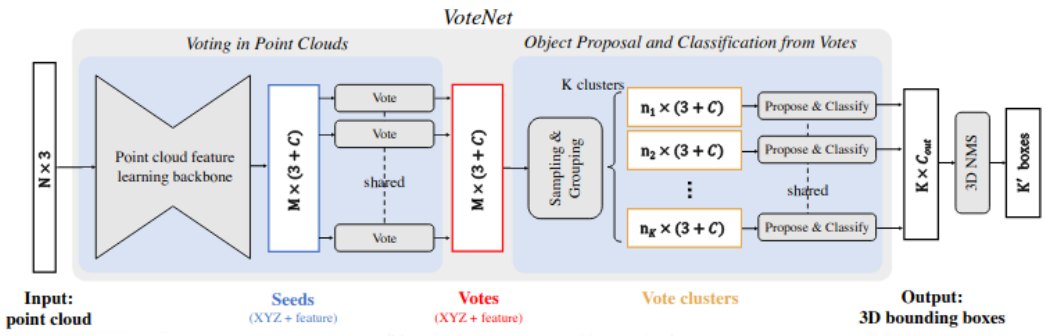


Figure 4.1: Structure of VoteNet

In the original implementation, the network was divided into two sections: the voting and object proposal. In this thesis, their implementation has been rearranged structuring the network into three sections.

1. The backbone: take the stixels and apply a first feature extraction.

2. The voter: group the feature in order to create a small cluster.

3. The detector: assign for each cluster a label and a position.

This has been done in order to have a better highlight to the backbone that is a fundamental part of the network.

In the following paragraphs, all these parts are deeply explained showing their structure and the change from the original implementation.

4.1 The backbone

The backbone takes as input the vector of stixels of size $N \times (3 + 2)$, for each N stixel, it is specified the 3D coordinate of the center of this stixel and his height and width. So each stixel can be thought like a 3D point (the center) and two adding features (the height and the width), this allows to feed VoteNet in a simpler way than a normal point cloud.

The goal of this section is Point cloud feature learning through a deep neural network without using any hand-crafted features.

The results of the backbone are $M \times (3 + C)$ seed point dimension. Each seed point has both a 3D coordinate and a high dimensional feature vector.

Using the recent knowledge on neural network, that works on point cloud the backbone used for feature learning is the one proposed in PointNet++. The authors of VoteNet choose this as backbone due to its simplicity and demonstrated success on tasks ranging from normal estimation, semantic segmentation to 3D object localization.

The backbone network has several set-abstraction layers (SA layer) and feature propagation (upsampling) layers (FP layer) with skip connections.

In particular, the network has three set abstractions (SA) layers and one feature propagation/up-sampling (FP) layer.

The output of the FP layer is one vector of M seed point and each seed point will generate one vote in the following part of the network.

4.1.1 The Set-Abstraction layers layer

The set abstraction layer has been proposed in PointNet++, its goal is to make a feature extraction, and taking a set of points it produces a new set with fewer elements.

The set abstraction level is made of three layers:

1. Sampling layer: selects a set of points from input points, which defines the centroids of local regions.

2. Grouping layer: constructs local region sets by finding "neighboring" points around the centroids.

3. PointNet layer: uses a mini-PointNet to encode local region patterns into feature vectors.

Each set abstraction level have as input a $N \times (d + C)$ matrix where each N points have a $d - dim$ coordinates and $C - dim$ point feature. In this case the d-dim is fixed a 3 and the C-dim change in each layer. The output of this layer is an $N' \times (d + C')$ matrix of N' subsampled points with $d - dim$ coordinates and new $C' - dim$ feature vectors summarizing local context.

Sampling layer

The first phase is to select the centroid which has to group the other points in the next phase. There are many ways to perform this action and they are iterative farthest point sampling (FPS) and Fusion Sampling (FS).

The farthest point sampling (FPS) is an algorithm that from a set of point $\{x_{i1}, x_{i2}, ..., x_{im}\}$ select a point x_{ij}, such that x_{ij} is the most distant point from the set $\{x_{i1}, x_{i2}, ..., x_{ij-1}\}$ with regard to the rest points.

The original algorithm selects a subset of point basing on metric distance. In order to improve the result, it has been also tried another type of grouping call Fusion Sampling (FS) proposed in 3DSSD. This select the output subset joins

two different subsets, the first part of the output is based on farthest point on metric distance, the second part on farthest point on feature distance.

Then after founding these centroids they are passed on to the Grouping layer.

Grouping layer

The input to this layer is a point set of size $N \times (d + C)$ and the coordinates of a set of centroids of size $N' \times d$ provide by the Sampling layer. The output are groups of point sets of size $N' \times K \times (d + C)$, each group corresponds to a local region, and K is the number of points in the neighborhood of centroid points. K can vary across the group due to the sparsity of the point cloud.

In a point set, the neighborhood of a point is defined by metric distance. The algorithm group all the points inside a sphere of a certain radius. Another way to grouping the point is through kNN, which selects the K Nearest neighbor of the centroid. This method is not explored because according to the authors of PointNet ++ fixing a radius help to make local region feature more generalize across space, which is preferred for tasks requiring local pattern recognition.

PointNet layer

In PointNet layer, the input are N' local regions of points with data size $N' \times K \times (d + C)$. In the output the local region are abstracted by its centroid and local feature that encodes the centroid's neighborhood. Output data size is $N' \times (d + C')$.

In a local region, the points are translated using as reference frame the centroid. This is simply done by subtracting the center of the centroid in each point that is inside the region.

Then for each point is applied PointNet in order to extract features capturing the point-to-point relations in the local region. In this implementation, the Multi Perception Layer proposed is a sequence of convolution with kernel one that works as a fully connected layer, a batch normalization, and a ReLU.

As said in chapters before a point cloud has a non-uniform density, so it

comes out that sometimes an object has a surface with a lot of points and sometimes not. A requirement of this network is to capture the reference and learned feature both when the points on surfaces are dense and not in order to recognize both fine-grained local structures and sparsely sampled regions. To achieve this requirement it has been proposed adaptive PointNet layers that learn to combine features from regions of different scales when the input sampling density changes. There are two types of combining way:

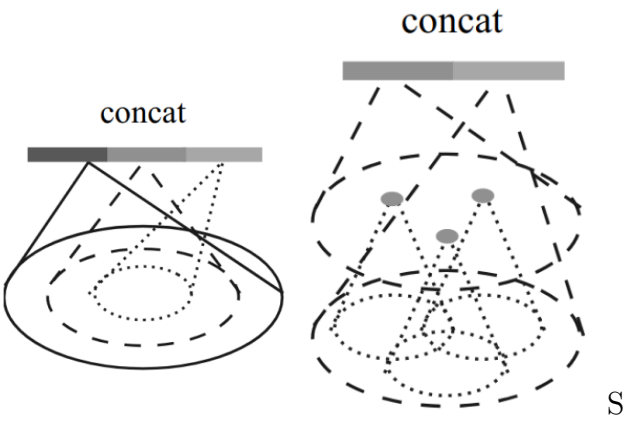
- Multi-scale grouping (MSG). a simple but effective way to capture multi-scale patterns is to apply grouping layers with different scales followed by PointNet to extract features of each scale. Features at different scales are concatenated to form a multi-scale feature.

- Multi-resolution grouping (MRG). with this grouping, the features of a region is a concatenation of two vectors. One vector is obtained by summarizing the features at each sub-region from the lower level L_{i-1} using the set abstraction level. The other vector (right) is the feature that is obtained by directly processing all raw points in the local region using a single PointNet. In this way, the first part is more reliable when the density is high, and the second part is more reliable when the density of the points in the region is low.

The grouping algorithms tested in this thesis are the grouping with one radius and the Multiscale grouping due to his simplicity and fast implementation.

4.1.2 The Feature Propagation layer

In the set-abstraction layer, the original point set is sub-sampled. This is due to the fact that the algorithm wants to concentrate the information on some more important points aggregating more features. This concentration causes also a loss of information in order to overcome that there is a feature propagation layer.



(a) Multi-scale grouping (MSG) (b) Multi-resolution grouping (MRG)

Figure 4.2: Grouping algorithm

In a feature propagation layer, the point features are propagated from $N_l \times (d + C)$ points to N_{l-1} points where N_{l-1} and N_l (with $N_l \leq N_{l-1}$) are point set size of input and output of set abstraction level l. The feature propagation is do interpolating feature values f of N_l points at coordinates of the N_{l-1} points. For each point of N_{l-1} it has been found the k closest point in N_l layer and then find the distance of this k closest point. This distance is used in the interpolation because it is used the inverse distance weighted average based on k nearest neighbors (usually $k = 3$). Then the interpolated features on N_{l-1} points are concatenated with of N_l points. The concatenated features are then passed through a small PointNet, which is composed of a one-by-one convolution in CNNs a batch normalization, and ReLU layers.

4.2 The voter

The idea of the voting is based on the concept of 2D Hough voting [22]. The 2D Hough voting is the first kind of 2D object detector composed of an offline and an online phase. In the offline phase, there is the creation of a code-book of local appearances that are characteristic for (a particular viewpoint of) its member objects. This is done by extracting local features around interest

points and grouping them with an agglomerative clustering scheme. So in the codebook, there are stored mappings between image patches(or their features) and their offsets to the corresponding object centers.

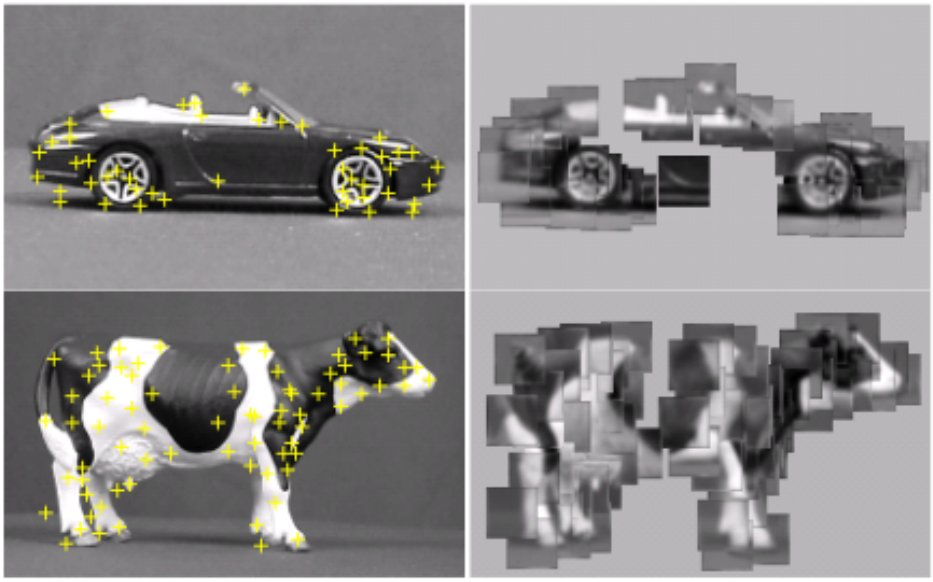


Figure 4.3: Example of feature extracted on coding book

In the online phase, interest points are selected from the image to extract patches around them. These patches are then compared against patches in the codebook to retrieve offsets and compute votes. As object patches will tend to vote in agreement, clusters will form near object centers. Finally, the object boundaries are retrieved by tracing cluster votes back to their generating patches.

The idea takes from this Hough voting is that the :

- voting-base detection that is more suitable for sparse point cloud than the region proposal network (RPN).
- bottom-up principle where small bits of partial information are accumulated to generate a confident detection.

In VoteNet this concept has been fused with the knowledge of neural networks, so interest feature points are described and selected by deep neural networks

instead of depending on hand-crafted features. The Vote generation and aggregation are also learned by a network instead of using a codebook. In this way, VoteNet is a single end-to-end train-able network named VoteNet.

This part of the neural network takes as input the centroid obtain from the second SA layer of the backbone and the feature extracting from the backbone. So it has a shape that is $M \times (3 + C)$ where M is the number of seed points, 3 is the three dimension of the centroid $x_i \in R^3$ and C is the number of feature $f_i \in R^C$.

The $M \times C$ feature are passing twice through a small point net composed of a 1D convolution, a batch normalization, and a ReLU.

Then the output of this PointNet is a vector that has shape $M \times (3 + C)$, this is used to generate an offset both for feature $\Delta f_i \in R^C$and for seed point $\Delta x_i \in R^3$ starting from the feature $f_i \in R^C$ extracting from the backbone.

This offset is adding both to the centroid and to the input feature of the Voter such that the vote $v_i = [y_i; g_i]$ where $y_i = x_i + \Delta x_i$ and $g_i = f_i + \Delta f_i$.

The predicted 3D offset Δx_i is explicitly supervised by a regression loss explain in the following paragraphs.

Votes have the same center as the centroid but their position is more close to the center than the original seeds. This increase the combination of the feature of different parts of the object in order to have an easier aggregation.

The second phase of the voter is the clustering. and this is done using a SA layer presented before. Starting from a set of votes $\{v_i = [y_i; g_i] \in R^{3+C}\}_{i=1}^{M}$, first it's sample a subset of K votes using farthest point sampling based on $\{y_i\}$ in 3D Euclidean space, to get $\{v_{ik}\}$ with $k = 1, ..., K$. After that K clusters are formed finding neighboring votes to each of the v_{ik}'s 3D location: $C_k = \{v_i^{(k)} \mid\mid\mid v_i - v_{ik} \mid\mid \leq r\}$ for $k = 1, ..., K$.

This clustering technique has been chosen due to its simplicity to integrate into an end-to-end pipeline.

Then the output of this clustering is passed to the detector that its goal is to find proposal and classification from vote clusters.

4.3 The detector

The detector is the last phase of the neural network and its goal is to extract the 3D bounding box starting from the feature extracting and aggregate from the voter.

It starts from a vote cluster that is in essence a set of high-dim points, so it is possible to leverage a generic point set learning network to aggregate the votes in order to generate object proposals. In VoteNet it has been used a shared PointNet.

So, starting from the feature generating by the vote aggregation it first passes the entire feature through an MLP composed of a 1D convolution, a batch normalization, and a ReLU. This extracted feature is put inside two different neural networks that one returns the class for each cluster and the other returns the size, position, and orientation of the bounding box.

4.4 Loss Function

One important part of the Neural network is the Loss because a network without a good loss cannot perform. This neural network has to perform many tasks so a multi-task loss has been designed.

The loss is composed of a voting loss, an objectness loss, a 3D bounding box estimation loss. In the original version, there is also a semantic classification loss but the goal of the neural network presented in this thesis is not the semantic classification, so, in order to perform in a better way, the more important loss has been deleted. The overall loss can be expressed as:

$$L_{VoteNet} = \lambda_1 L_{vote-reg} + \lambda_2 L_{obj-cls} + \lambda_3 L_{box}$$

Where the losses has been weightd in order to be in similar scale so $\lambda_1 = 1$, $\lambda_2 = 0.5$, $\lambda_3 = 1$.

Vote regression Loss

The 3D offset Δx_i predicted by the first MPL layer of the Voter is explicitly supervised by a regression loss:

$$L_{vote-reg} = \frac{1}{M_{pos}} \sum_i \parallel \Delta x_i - \Delta x_i^* \parallel 1[s_i \, on \, object]$$

where $1[s_i \, on \, object]$ indicates whether a seed point s_i is inside an annotated bounding box and M_{pos} is the count of total number of seeds on object surface. Δx_i^* is the ground truth displacement from the seed position x_i to the bounding box center of the object it belongs to. In cases that a point is in multiple ground truth boxes, it has been kept a set of up to three ground truth votes, and consider the minimum distance between the predicted vote and any ground truth vote in the set during vote regression on this point.

This loss can be easily thought of as a Mean Absolute Error, or L1 loss.

Objectness scores

The objectness loss is a cross-entropy loss for two classes. The cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverge from the actual label. So predicting a probability of 0.012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

As the predicted probability approaches 1, log loss slowly decreases. As the predicted probability decreases, however, the log loss increases rapidly. Log loss penalizes both types of errors, but especially those predictions that are confident and wrong.

In binary classification, where the number of classes M=2 cross-entropy can be calculated as:

$$-(y \log(p) + (1 - y) \log(1 - p))$$

If M>2 (example in multi-class classification), first there is calculated a separate loss for each class label per observation and then sum the result.

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

Where y is binary indicator (0 or 1) if class label c is the correct classification for observation o. p is the predicted probability observation o is of class c.

Box loss

The box loss is composed of center regression, heading estimation, and size estimation sub-losses. It has been derived by the one proposed in frustum PointNet [11].

$$L_{box} = L_{center-reg} + 0.1L_{angle-cls} + L_{angle-reg} + L_{corner} + L_{size-reg}$$

In all regression in the box loss, we use the robust L1-smooth loss. Both the box and semantic losses are only computed on positive vote clusters and normalized by the number of positive clusters.

$L_{center-reg}$ is for center regression of box estimation net; $L_{angle-cls}$ and $L_{angle-reg}$ are losses for heading angle prediction while L_{corner} and $L_{size-reg}$ are for box size.

In frustum the proposed L_{corner} and it has been thought in order to have a better balance when the center and size are accurately predicted, but the heading angle is off. Without this loss, a normal loss will penalize also the predicted center and the size of the box. In order to overcome this issue, it has been added L_{corner} and it is formalized in this way:

$$L_{corner} = \sum_{i=1}^{NS} \sum_{j=1}^{NH} \delta_{ij} min\Big\{ \sum_{k=1}^{8} \parallel P_k^{ij} - P_k^* \parallel, \sum_{i=1}^{8} \parallel P_k^{ij} - P^{**} t_k \parallel \Big\}$$

The corner loss is the sum of the distances between the eight corners of

a predicted box and a ground truth box. Since corner positions are jointly determined by center, size, and heading, the corner loss is able to regularize the multi-task training for those parameters. It has been designed firstly constructing a $NS \times NH$ "anchor" boxes from all size templates and heading angle bins. The anchor boxes are then translated to the estimated box center.

The anchor box corners are denoted as P_k^{ij}, where i, j, k are indices for the size class, heading class, and (predefined) corner order, respectively. To avoid large penalty from flipped heading estimation, distances to corners are we further computed (P_k^{**}) from the flipped ground truth box and use the minimum of the original and flipped cases. δ_{ij}, which is one for the ground truth size/heading class and zero else wise, is a two-dimensional mask used to select the distance term we care about.

One difference from Frustum is that, instead of a naive regression loss, it has been used a Chamfer loss [23] for $L_{center-reg}$ (between regressed centers and ground truth box centers).

The Chamber loss is defined as Chamber distance that is the distance between $S_1, S_2 \subseteq R^3$ as:

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

d_{CD} is not a distance function because triangle inequality does not hold. For each point, the algorithm of CD finds the nearest neighbor in the other set and sums the squared distances up. Viewed as a function of point locations in S_1 and S_2 , CD is continuous and piece-wise smooth. The range search for each point is independent, so it is easy to parallelize.

Chapter 5

Results

One complexity of the neural network is that there are too many parameters and too many ways to tune it. The tuning has to do with a freeway, leading by intuition and personal experience. The most important thing is to record all the results in order to have a better view of which path takes.

In order to compare all the results, there are common evaluation metrics that change from dataset to dataset. In this case, the dataset used is Kitti its specific metrics for the evaluation are based on the Average Precision (AP). That is explained in the first part of this chapter.

The second part is dedicated to explain and analyze the experiments. In order to have a better overview, many paths have been taken and some have improved the results and others not. In this part has been reported the main trials analyzing the advantages and the disadvantages of each setup.

In the third part of this chapter, there is a comparison between the other Neural Networks that represent the state of the art in 3D object detection. The normal test bench for the comparison is the Kitti test set. In this thesis has been used only the evaluation split of the training set. This has been due to the fact that it is difficult to be accepted to analyze the result of the training on the test set that has been done by Kitti creators. Despite this problem, the split of the training set can be adopted as a base for the comparison.

5.1 Evaluation metrics

The evaluation metrics adopted in this thesis is the one defined by Kitti dataset [1] for 3D object detection. This metrics is average precision and this concept is used to validate many parameters.

A key element for the Average precision is the Intersection over Union (IoU). In 2D the IoU measures the overlap between two boundaries. It is used to measure how much our predicted boundary overlaps with the ground truth boundary (the real object boundary).

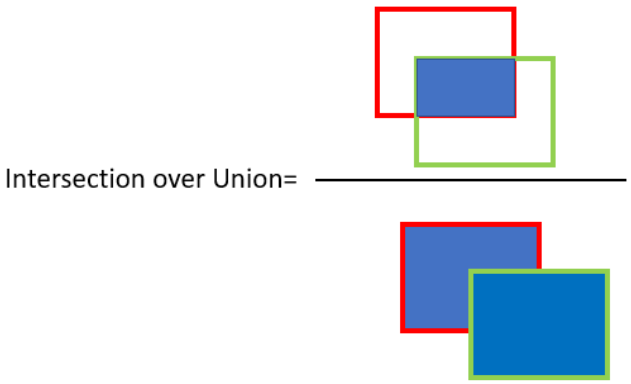


Figure 5.1: 2D Intersection over Unit representation.

The Intersection over Unions tells how much the predicted bounding box and the ground truth bounding box overlap. The same can be applied to the 3D case. In this case, it is not referred to areas but to volumes, but the concept can be kept as the same.

So the Average Precision is a measure that combines recall and precision for ranked retrieval results.

The Precision (also called positive predictive value) measures how accurate are your predictions. (i.e. the percentage of your predictions are correct). It can be seen as the fraction of relevant instances among the retrieved instances.

Recall (also known as sensitivity) measures how good you find all the positives results. It can be seen as the fraction of the total amount of relevant instances that were actually retrieved.

Their mathematical definitions are:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Let's say we set IoU to 0.7, in that case:

- If IoU ≥ 0.7, classify the object detection as True Positive(TP);
- If IoU < 0.7, then it is a wrong detection and classifies it as False Positive(FP);
- When ground truth is present in the image and the model failed to detect the object, we classify it as False Negative(FN);
- True Negative (TN): TN is every part of the image where we did not predict an object. This metrics is not useful for object detection, hence we ignore TN.

So the formula for the Average Precision is:

$$AP = \frac{1}{GTP} \sum_{k}^{n} Precision_k \times rel_k$$

Where GTP refers to the total number of ground truth positives, refers to the precision, and rel_k is a relevance function. The relevance function is an indicator function which equals 1 if the result at rank k is relevant and equals 0 otherwise.

Another metric use is the Average Heading Similarity (AHS). The AHS is the Average Orientation Similarity (AOS) but evaluated using 3D IOU and global orientation angle instead of 2D IOU and observation angle. The AOS can be defined as:

$$AOS = \frac{1}{11} \sum_{r \in \{0, 0.1, .., 1\}} \max s(\widetilde{r})_{\widetilde{r}\,:\,\widetilde{r} \geq r}$$

r is the recall and is define as $r = \frac{TP}{TP+FN}$ where detected 2D bounding boxes are correct if they over-lap by at least 50% with a ground truth bounding box. s is the orientation similarity $s \in [0, 1]$ at recall r is a normalized ($[0..1]$) variant of the cosine similarity defined as:

$$s(r) = \frac{1}{|D(r)|} \sum_{i \in D(r)} \frac{1 + \cos \Delta\theta^{(i)}}{2} \delta_i$$

where $D(r)$ denotes the set of all object detection at recall rate r and $\Delta_\theta^{(i)}$ is the difference in angle between estimated and ground truth orientation of detection i. To penalize multiple detection which explain a single object, it has been set $\delta_i = 1$ if detection i has been assigned to a ground truth bounding box (overlaps by at least 50) and $\delta_i = 0$ if it has not been assigned.

The change from AOS to AHS has been made in order to, removing the metric's dependence on localization accuracy.

In Kitti evaluation the metrics are:

- car detection AP: 2D Average Precision based on the bounding box projected on cam 2 of Kitti;
- car detection BEV AP: 2D Average Precision based on the bounding box projected Bird-eye view;
- car orientation BEV AHS. AHS orientation of the bounding box projected Bird-eye view;
- car detection 3D AP: 3D Average Precision based on the 3D bounding box;
- car orientation 3D AHS: AHS orientation of the 3D bounding box.

These evaluation metrics have been used on the evaluation test bench. The evaluation test bench is a partition of the training set and this partition has been created as a standard in order to have most of the type of Kitti scenes.

5.2 Experiment

Due to the innovation of the type of data used for 3D object detection many experiments have been done in order to find a better balance of the parameters. The starting point of this experiment was the and adapted VoteNet that was trained to work on the point cloud data provided by the Kitti dataset.

The tables 5.1, 5.2, 5.3 show the starting configuration of the VoteNet.

Table 5.1: Backbone Set Abstraction layer

parameters	SA 1	SA 2	SA 3
number seed point	4096	512	256
number of grouping point for each seed point	64	32	32
radius of the grouping seed point	0.4	1.6	3.2
Multi layer perception level	[32,32,64]	[64, 96, 128]	[128, 256, 256]
sampling type	D-FPS	D-FPS	D-FPS

Table 5.2: Backbone Feature propagation layer

parameters	FP
number of grouping point for each seed point	384
Multi layer perception level	[256, 256]

Table 5.3: Voter Set Abstraction layer

parameters	SA Detector
number seed point	256
number of grouping point for each seed point	16
radius of the grouping seed point	0.75
Multi layer perception level	[256, 256, 512]
sampling type	D-FPS

Starting from this configuration the main changes have been done in the Set Abstraction layer present on the backbone in order to adapt it to the new type of data. The other layer has not to be modified because several tests confirm that is the right setting.

5.2.1 Stixels as points

A first test was to consider the stixels as the combination of four 3D points that represent the four vertices and then feed the network with this type of data. Even if the data that feeds the network are points without any characteristics, they are also derived from medium type data. They are not simple points that represent the whole scene but only the vertical parts, so there is also a reduction of points used.

The use of the network for these early trials is the original VoteNet. The first change that was made was to set the batch size of the training. The size of the batch under investment was 4 and 8.

Table 5.4: Test 0: batch size 4; Test 1: batch size 8

	test 0			test 1		
metrics	easy	medium	hard	easy	medium	hard
AP	87.58	76.12	73.54	85.18	72.82	67.22
BEV AP	84.39	71.01	66.83	81.07	69.04	64.34
BEV AHS	82.81	69.37	65.15	79.91	67.43	62.73
3D AP	70.17	55.10	53.63	65.55	51.57	49.20
3D AHS	68.88	54.05	52.57	64.73	50.74	48.35

Analyzing the accuracy of the test reported table 5.4 with batch size four for test 0 and batch size eight in test 1 it comes out that the test with a smaller batch size has a better result. This has been confirmed also from other further tests that are not reported. According to *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*[24] this can be derived that

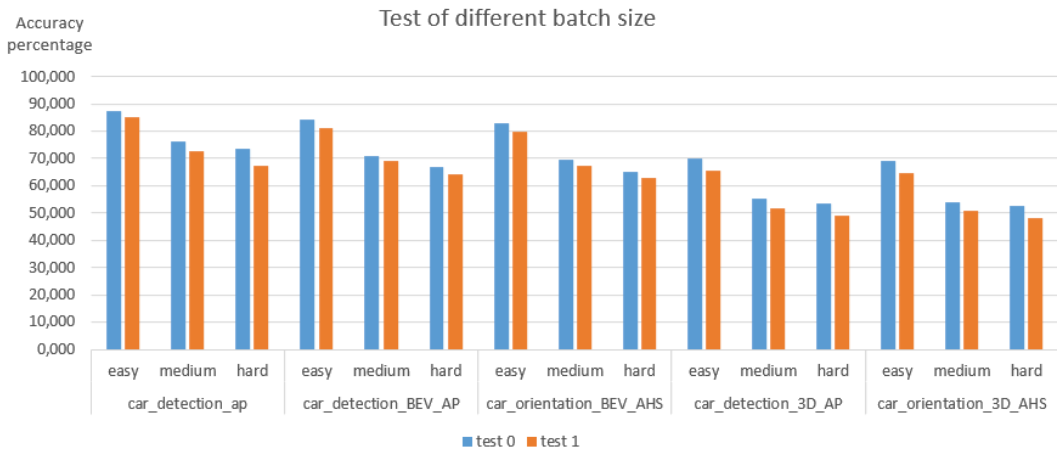


Figure 5.2: Test of different batch size

using a larger batch there is a significant degradation in the quality of the model, as measured by its ability to generalize.

5.2.2 Stixels as stixels

After that initial trial, it starts to consider the stixel not only with the composition of four points that summarize it but as one point that corresponds to the center of the stixel and two feature that corresponds to the height and width of the stixel. This has been done in order to fully exploit the properties of stixel reducing the entire point set size at max 3200. This number can be further reduced by deleting the outliers explained in chapter 3.

Table 5.5: Test 0: stixel 4 points; Test 4: stixel one point

	test 0			test 4		
metrics	easy	medium	hard	easy	medium	hard
AP	87.58	76.12	73.54	87.66	76.63	74.70
BEV AP	84.39	71.01	66.83	85.14	73.59	67.76
BEV AHS	82.81	69.37	65.15	84.02	71.94	66.10
3D AP	70.17	55.10	53.63	70.91	58.34	54.56
3D AHS	68.88	54.05	52.57	70.04	57.32	53.47

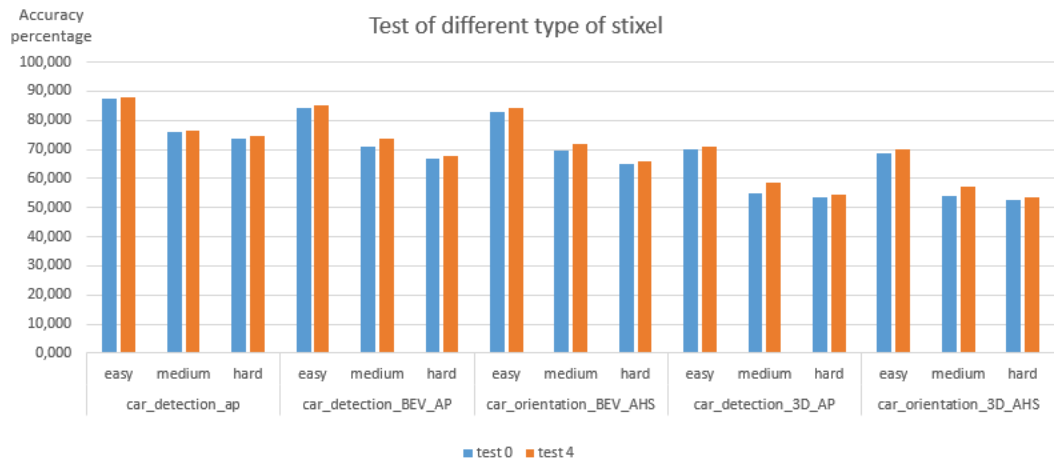


Figure 5.3: Test of different type of stixel

Despite using a reduced type of data results that the network has a better behavior increasing a little bit the accuracy in particular for the 3D bounding box and for the estimation of the pose in bird-eye of view. This is possible to see in table 5.5 where test 0 is the previous test in which the stixel was represented like four points and test 4 where the stixel were represented like a single point with high and width as a feature. This is an important result because it underlines that having a medium type data allows achieving better accuracy with a number of points that is twenty-two times less than the original dataset.

Thanks to this result, all the next trials have been made considering the stixel as a center point, that has as a feature the height and the width of that stixel.

5.2.3 Change the type of Sampling layer

A third trial done was to change the sampling layer in the set abstraction layer. This has been done to modify the configuration of VoteNet in order to extract the seed point not only based on the euclidean distance but also combine the feature distance. This change has been made in order to add a degree of freedom in the sampling layer according to 3DSSD [10].

Other changes involve are the reduction of the number of seed points extracted

from this in the sampling layer. This change has been made in order to adapt the network for the different types of sampling layers and adjust the data according to the reduced number of input points. A further explanation has been derived in the following paragraphs.

Table 5.6: Backbone Set Abstraction: Change the type of Sampling layer

	test 4			test 6		
parameters	SA 1	SA 2	SA 3	SA 1	SA 2	SA 3
seed point	4096	512	256	4096	256	128
grouping point	64	32	32	64	32	32
radius	0.4	1.6	3.2	0.4	1.6	3.2
MLP	32,32,64	64,96,128	128,256,256	32,32,64	64,96,128	128,256,256
sampling	D-FPS	FS	FS	D-FPS	FS	FS

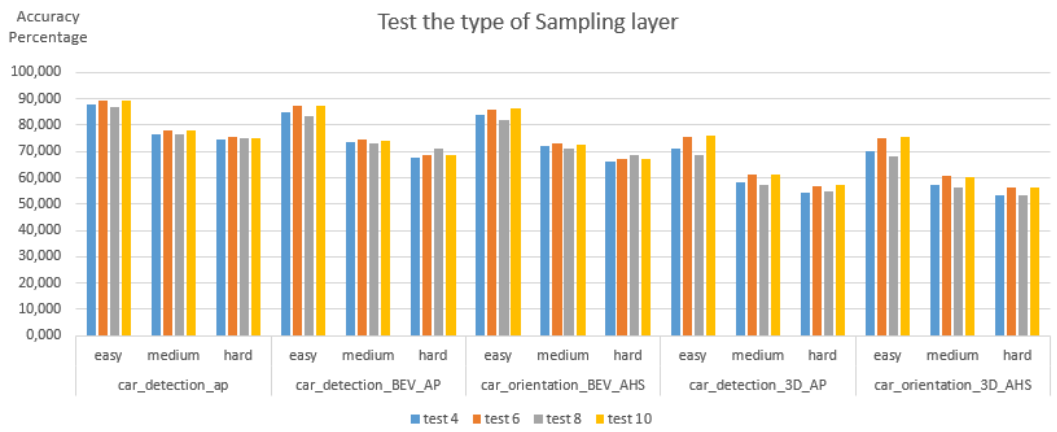


Figure 5.4: Test the type of Sampling layer

Analyzing the result it has emerged that changing the sampling layer is very important in order to improve the performance.

This can be seen in test 10 where both in the second and the third layer it has been applied the FS sampling layer.

Combining both Euclidean distance and feature distance it is possible to achieve a better result more than use one type of sampling layer. This can

	test 8			test 10		
parameters	SA 1	SA 2	SA 3	SA 1	SA 2	SA 3
seed point	2048	512	256	2048	256	128
grouping point	64	32	32	64	32	32
radius	0.4	1.6	3.2	0.4	1.6	3.2
MLP	32,32,64	64,96,128	128,256,256	32,32,64	64,96,128	128,256,256
sampling	D-FPS	D-FPS	D-FPS	D-FPS	FS	FS

	voter	
parameters	SA	SA
seed point	256	256
grouping point	16	16
radius	0.75	0.75
MLP	256,256,512	256,256,512
sampling	D-FPS	F-FPS

Table 5.7: Test the type of Sampling layer

	test 4			test 6		
metrics	easy	medium	hard	easy	medium	hard
AP	87.66	76.62	73.70	89.32	78.22	75.34
BEV AP	85.14	73.59	67.76	87.19	74.73	68.54
BEV AHS	84.02	71.94	66.10	86.14	73.29	67.15
3D AP	70.91	58.34	54.55	75.64	91.36	56.94
3D AHS	74.04	57.32	53.46	74.90	60.50	56.08

be derived due to the sparsity of the starting point cloud extracting from an outdoor environment. In that type of environment, the point cloud is less concentrated so the point (or stixel) that constitutes an object can be so far that can be confused with the background. The feature helps to discriminate the background and the object point increasing the accuracy of the network.

Table 5.8: Test the type of Sampling layer

	test 8			test 10		
metrics	easy	medium	hard	easy	medium	hard
AP	86.80	76.65	75.19	89.49	77.78	75.02
BEV AP	83.23	72.84	70.88	82.52	70.52	68.61
BEV AHS	82.08	70.86	68.69	86.41	72.49	67.25
3D AP	68.77	57.42	54.58	76.09	61.21	57.23
3D AHS	68.00	56.22	53.38	75.51	60.38	56.40

Decreasing the number of seed points also increases the accuracy of the network. This can be derived from the reduced number of initial points that need fewer seed points. If there are too many seed points there is an overestimation of the number of the centers, so it may derive too much false positive decreasing the accuracy of the network. This behavior is better analyzing in the following paragraphs.

5.2.4 Data augmentation

An important improvement on the accuracy of the network has been obtained adding the data augmentation on training data. Data augmentation are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularize and helps reduce overfitting when training a machine learning model.

The type of data augmentation used are:

- Global flip: Randomly mirroring the stixel and boxes using as mirror plane the plane y-z of the camera;
- Global rotation: Rotate the entire scene around the y-axis of the camera;
- Local rotation: Rotate each car with a random rotation around each own vertical axis;

- Local translation: Translate each car having as center each own vertical axis.

An important operation was the tuning of the range where this random movement can be selected. This was important because the stixels don't constitute the entire shape of a car but only the visible part from the camera and the LIDAR, so if the range of the rotation and the translation are too big they create a non-real combination of stixel that describe a shape of the vehicle.

The last augmentation adopted is the Mixup augmentation. It consists of the adding of the stixels of extra vehicle extracted from the other scenes. This is used to enrich the scene in order to give the network more examples to learn.

The mixup augmentation is composed of two phases:

- Offline phase: where it is created a codebook containing the information of all the vehicle of the scene and their stixel that compose that vehicle;
- Online phase: where randomly it has added to the scene some vehicles.

The policies for the enrichment of the scene were added vehicles until it reaches the maximum number of the vehicles, taking into account also the vehicles that were already present in the scene.

After this adding phase, the new vehicle also has an augmentation, and then there is a resolution of conflict. It can happen that adding a vehicle collides with another vehicle already present so in order to avoid this, there is a collision resolution phase where the vehicle that creates conflicts are deleted and, if it is possible, keeping the vehicles that were already present in the scene before the mixup augmentation.

In order to have a better comparison of the result of a different kind of data augmentation, it has been chosen to start from one single network and then apply in the training phase the different types of data augmentation.

Table 5.9 show the starting network and table 5.10 the legend for the various kind of test and which test ID is connected.

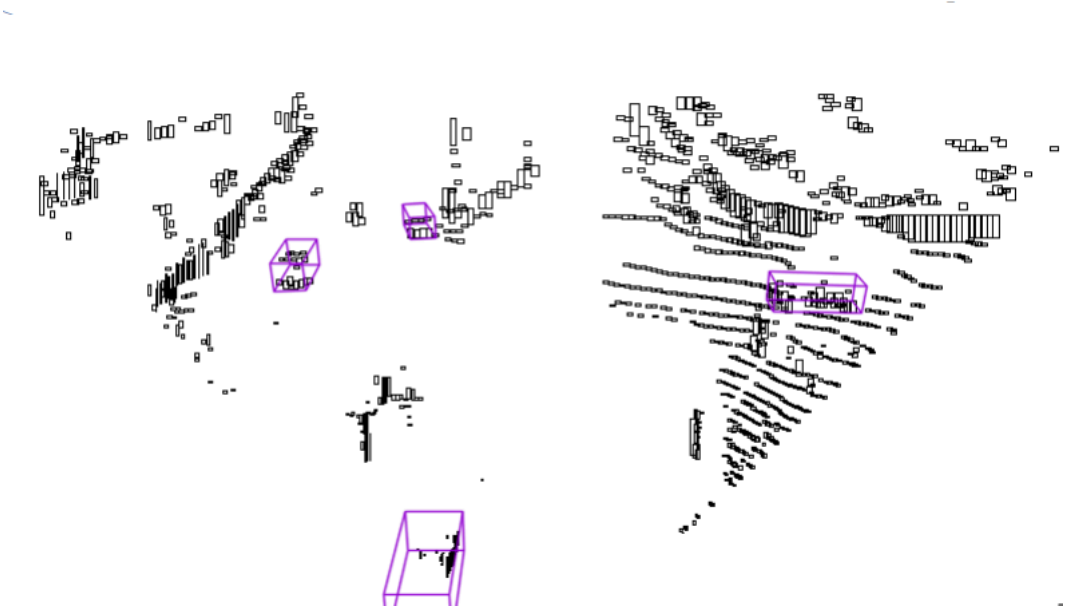


Figure 5.5: Scene without data augmentation

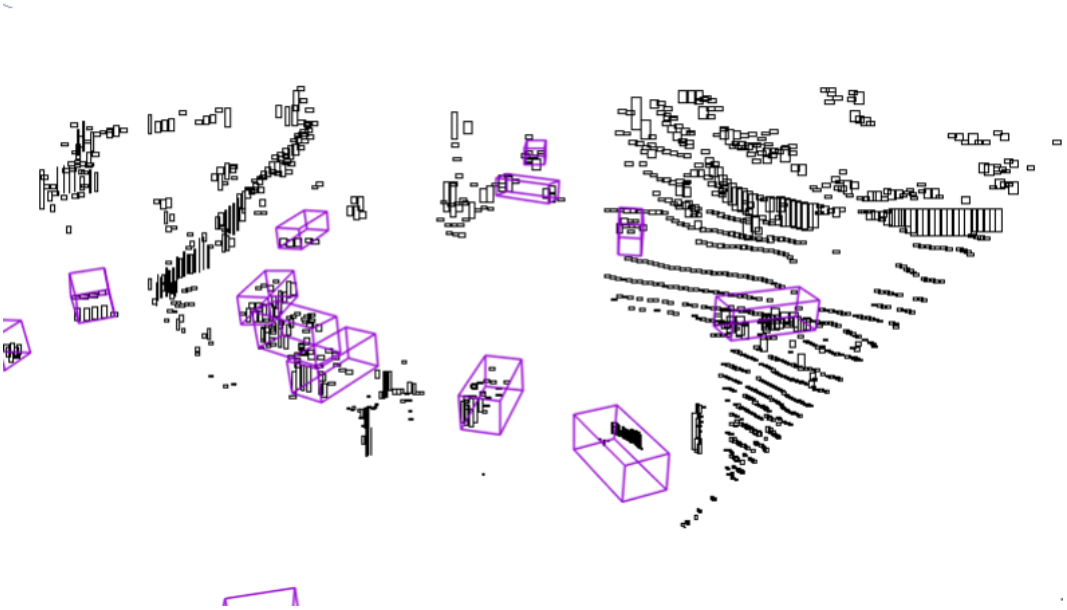


Figure 5.6: Scene with data augmentation

The augmentation provides an improvement to the result of the neural network increasing all the accuracy metrics.

This can be notice analyzing the result in 5.11 and 5.12.

The improvement is close to 10% and this is really relevant compared to the other improvement obtained in other tests. It can be derived from the fact that

Table 5.9: Data augmentation: starting network

	backbone			voter
parameters	SA 1	SA 2	SA 3	SA
seed point	2048	256	128	256
grouping point	64	32	32	16
radius	0.4	1.6	3.2	0.75
MLP	32,32,64	64,96,128	128,256,256	256,256,512
sampling	D-FPS	FS	FS	F-FPS

Table 5.10: Type of data augmentation test

test	augmentation	range
16	flip	prob. flip 0.5
17	global rotation	±10 degree
18	local rotation	±30 degree
19	local transition	±2 meters
20	flip, local rotation, local transition	
22	flip, local rotation, local transition, mixup	

Table 5.11: Test the type of data augmentation

	test 16			test 17			test 18		
metrics	easy	med.	hard	easy	med.	hard	easy	med.	hard
AP	89.45	78.81	77.54	60.69	51.95	52.15	90.07	77.10	73.17
BEV AP	88.24	76.80	73.25	52.37	44.16	41.47	89.36	74.32	67.96
BEV AHS	87.8	75.99	72.28	52.01	43.33	40.69	88.52	73.04	66.66
3D AP	76.80	65.43	61.67	30.11	23.78	22.88	80.20	62.63	57.05
3D AHS	76.56	64.90	61.07	29.67	23.36	22.51	79.45	61.72	56.12

the increase of the variety of the object increases the number of scenes in the training obtaining a better generalization.

Table 5.12: Test the type of data augmentation

	test 19			test 20			test 22		
metrics	easy	med.	hard	easy	med.	hard	easy	med.	hard
AP	89.96	79.15	76.82	90.13	79.23	76.07	90.33	82.23	78.20
BEV AP	88.66	77.43	72.95	89.29	77.43	72.89	89.94	79.09	76.52
BEV AHS	87.96	76.33	71.75	88.97	76.66	71.94	89.89	78.80	76.13
3D AP	80.80	66.02	61.50	83.16	66.99	62.06	88.18	69.51	67.48
3D AHS	80.16	65.31	60.75	82.94	66.55	61.55	88.13	69.35	67.25

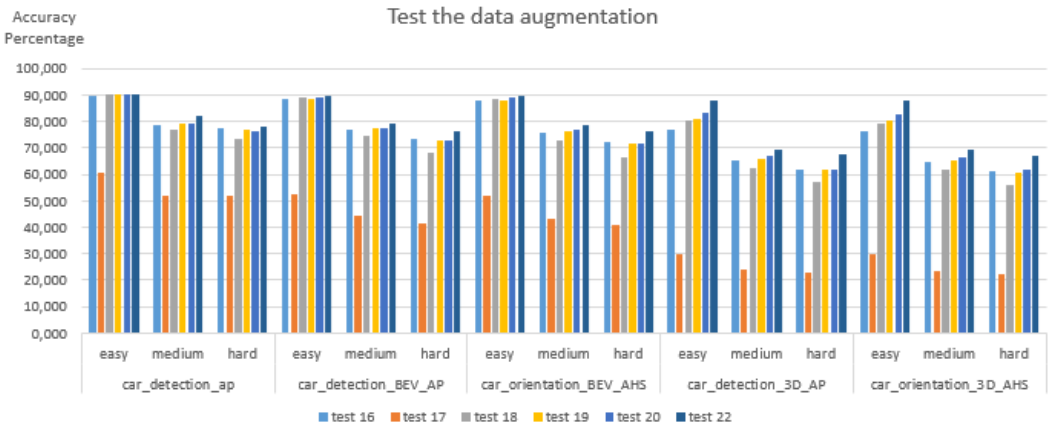


Figure 5.7: Test the data augmentation

An important constrain of the augmentation is that the scene cannot change too much otherwise the performance start to degrade has happened using the global rotation. As is possible to see in the graph the global rotation in test 17 decreases the accuracy this could happen because it changes too much the entire scene.

So important parameters to tune so are the max range of the augmentations: it cannot be too small because otherwise there the scene remains more or less the same, but it cannot change too much otherwise the network start to degraded the performance.

After these important results, the data augmentation is applied in all the next test, and in particular, the augmentation apply are :

- mixup augmentation;

- horizontal flip, prob. flip: 50%;
- local rotation: ±10 degrees;
- local shift: ±2 meters.

5.2.5 Reduction of number of seed points

After finding the better batch size, the type of augmentation, and the kind of sampling layer now this section of the test is focused on finding the better balance of the number of seed points extracted in the SA layer of the backbone.

Compared to the initial number of seed points now this number has been reduced. This reduction has been made in order to improve the performance of the network because the initial number of seed points was designed to a higher number of input points. Having as input at least 3000 points it is useless having a number of seed points that is 4096 because this means that there is more center than points.

A reduction of seed points should allow the network to focusing only on the most important points creating a better grouping without "disorientate" the network.

This reduction has been made until the network start to degraded his performance and it has been continued in order to understand the degradation of the performance at the decreasing of the number of the seed points.

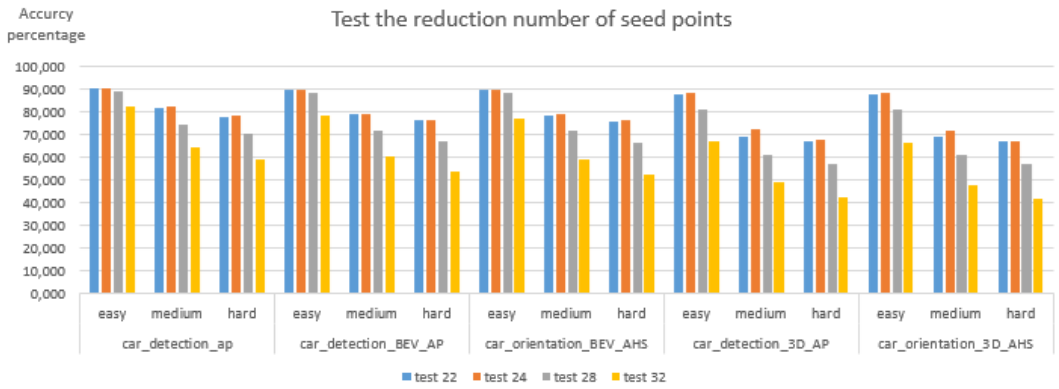


Figure 5.8: Test the reduction number of seed points

Table 5.13: Backbone Set Abstraction: Reduction of number of seed points

	test 22			test 24		
parameters	SA 1	SA 2	SA 3	SA 1	SA 2	SA 3
seed point	2048	256	128	1024	256	128
grouping point	64	32	32	64	32	32
radius	0.4	1.6	3.2	0.4	1.6	3.2
MLP	32,32,64	64,96,128	128,256,256	32,32,64	64,96,128	128,256,256
sampling	FS	FS	FS	FS	FS	FS

Table 5.14: Backbone Set Abstraction: Reduction of number of seed points

	test 28			test 32		
parameters	SA 1	SA 2	SA 3	SA 1	SA 2	SA 3
seed point	512	128	64	256	128	64
grouping point	64	32	32	64	32	32
radius	0.4	1.6	3.2	0.4	2.0	4.0
MLP	32,32,64	64,96,128	128,256,256	32,32,64	64,96,128	128,256,256
sampling	FS	FS	FS	FS	FS	FS

Table 5.15: Test the reduction of number of seed points

	test 22			test 24		
metrics	easy	med.	hard	easy	med.	hard
AP	90.33	82.23	78.20	90.56	82.71	78.42
BEV AP	89.94	79.09	76.52	90.24	79.4	76.89
BEV AHS	89.89	78.80	76.13	90.18	79.21	76.52
3D AP	88.18	69.51	67.48	88.60	72.25	67.64
3D AHS	88.13	69.35	67.25	88.55	72.09	67.44

This analysis discovers the best trade off of the number of seed points in each set abstraction level. The configuration of test 24 achieves a very good

Table 5.16: Test the reduction of number of seed points

	test 28			test 32		
metrics	easy	med.	hard	easy	med.	hard
AP	89.30	74.74	70.64	82.88	64.74	58.93
BEV AP	88.56	72.15	67.13	78.30	60.37	53.84
BEV AHS	88.36	71.60	66.47	76.99	58.94	52.34
3D AP	81.29	61.50	57.49	67.14	48.90	42.38
3D AHS	81.14	61.23	57.14	66.23	48.01	41.51

result. After that reducing the number of seed points there is a degradation of the performance. This can be derived to the fact that if the number of seed point is low one seed point have to group many points and the network is not able to capture all the details increase the possibility of errors.

Another goal of this test was to verified if reducing the seed points there is some reduction of memory occupy from the weights and some reduction of inference time. This tried has to be done in order to find a light neural network that is important in the next production phase.

The result discards this idea because reducing the number of seed point the memory occupies for the weights has been kept stable at 9.35Mb and also the inference time not change and it is fixed around 30 ms.

5.2.6 Multi-scale grouping

A characteristic of the point cloud in the outdoor environment is the sparsity. In the same scene, it is possible to find areas where there are many points concentrates and areas where there are fewer points. These areas can be also in the same object so the network should be able to understand both fine-grained local structures and sparsely sampled regions. The same concept can be applied to scene composition with stixel.

In order to overcome this problem, it has been tried two kinds of grouping layers. The first is the simple grouping layer where for each SA layer there is

only one radius. This has been tested in all the previous tests.

The second type tested is the multi-scale grouping where in each sampling layer there are many radii to group the points. Due to his simplicity of implementation, the grouping applies in this thesis is the Multi-scale grouping (MSG). MSG is a simple but effective way to capture multi-scale patterns is to apply grouping layers with different scales followed by PointNet to extract features of each scale. Features at different scales are concatenated to form a multi-scale feature.

The degrees of freedom that has to be tune in these tests for each set abstraction layer are:

- The size of radii of the grouping layer;
- the size of the network for each sampling;
- the number of final points.

Table 5.17: Test 36: Multi-scale grouping

	backbone		
parameters	SA 1	SA 2	SA 3
seed point	1024	256	128
grouping point	64	32	32
radius	0.4 0.8	1.6 2.0	3.2 4.0
MLP	32,32,64 32,32,64	64,96,128 64,96,128	128,256,256 128,256,256
sampling	FS	FS	FS

As it is possible to see in table 5.18 the use of the multi-scale grouping improve the result of the network. This confirmed that the sparsity of the point cloud is more suitable for multi-scale grouping.

According to the many tests not cited it is possible to see that a symmetric MLP of the SA layer allow achieving better results but the change are very small

Table 5.18: Test the Multi-scale grouping

	test 24			test 36		
metrics	easy	med.	hard	easy	med.	hard
AP	90.56	82.71	78.42	95.63	86.92	83.03
BEV AP	90.24	79.4	76.89	90.39	84.74	79.14
BEV AHS	90.18	79.21	76.52	90.07	84.49	78.88
3D AP	88.60	72.25	67.64	89.05	76.88	72.35
3D AHS	88.55	72.09	67.44	89.05	76.88	72.35

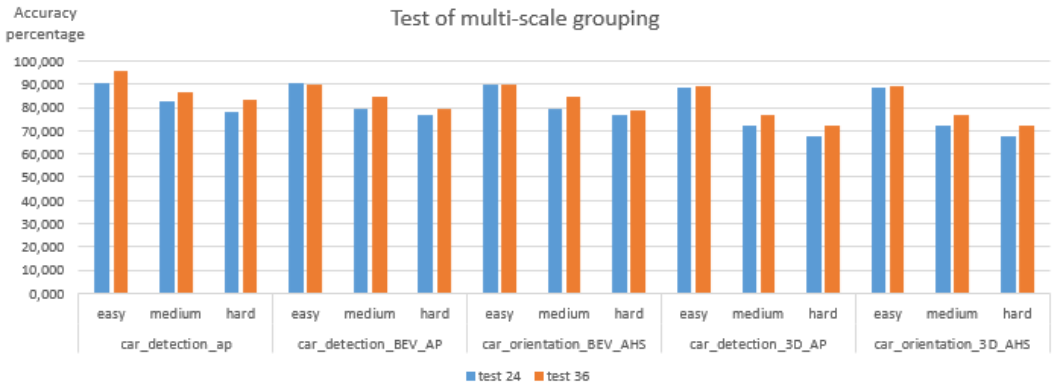


Figure 5.9: Test of multi-scale grouping

compared to the whole results and this can be due to the normal fluctuation in the training phase.

A drawback of this setup is the increasing of the memory occupied by the weight, which passes from 8.5Mb of the standard configuration to 14Mb. The inference time also increases from 30 ms to 50 ms.

Despite the drawbacks thanks to multi-scale grouping, the network has been able to achieve the state of the art.

5.2.7 Change the width of stixels

The last group of tests based on stixel generate from point cloud has been focused on changing the size width of the stixel. The goal of this test is to verify if having a group of stixel that can potentially have a better resolution

increase the accuracy of the network.

The network setup with this test has been evaluated was the one on test 24 and test 36. This two network has been choose because have allowed achieving the best result with and without multi-scale grouping.

The test is to reduce the size of the width of the cells of the matrix from 8 to 4 pixels.

With this configuration of stixel the average double from the previous configuration:

- average: 1996;
- standard deviation: 664;

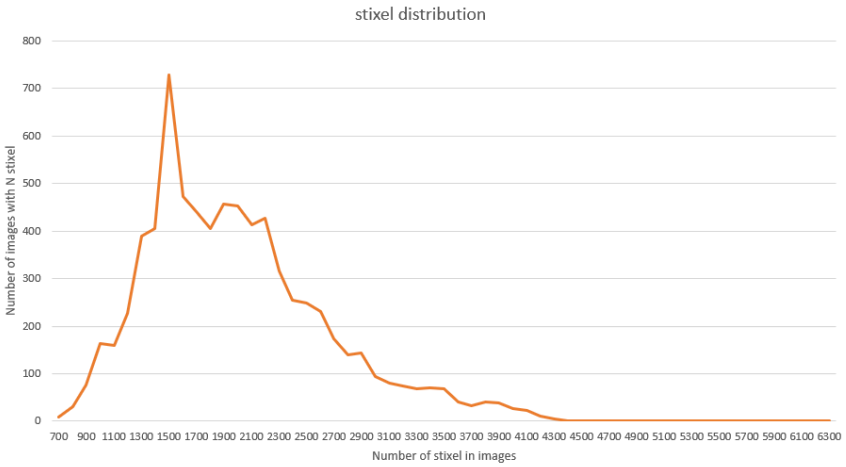


Figure 5.10: Distribution of stixel generate from reduced width stixel

As is possible to see in table 5.19 despite the doubling of the number of stixel the precision not increase. This means that with the right resolution of stixel it is possible to achieve very good results reducing also the global number of stixel.

The configuration of the network adopted in test 40 is the configuration of test 24 and the configuration adopted in test 42 is the configuration of test 36.

In other tests that have not been reported it has increased the number of seed points in order to adapt this number to the increasing number of stixel.

Table 5.19: Test of change width of stixel

	test 40			test 42		
metrics	easy	med.	hard	easy	med.	hard
AP	92.63	87.33	83.90	96.07	88.46	86.87
BEV AP	90.12	85.65	82.57	89.97	87.51	84.23
BEV AHS	90.08	85.36	82.26	89.95	87.28	83.22
3D AP	88.91	77.79	72.65	88.67	78.25	76.44
3D AHS	88.87	77.62	72.45	88.66	78.11	76.25

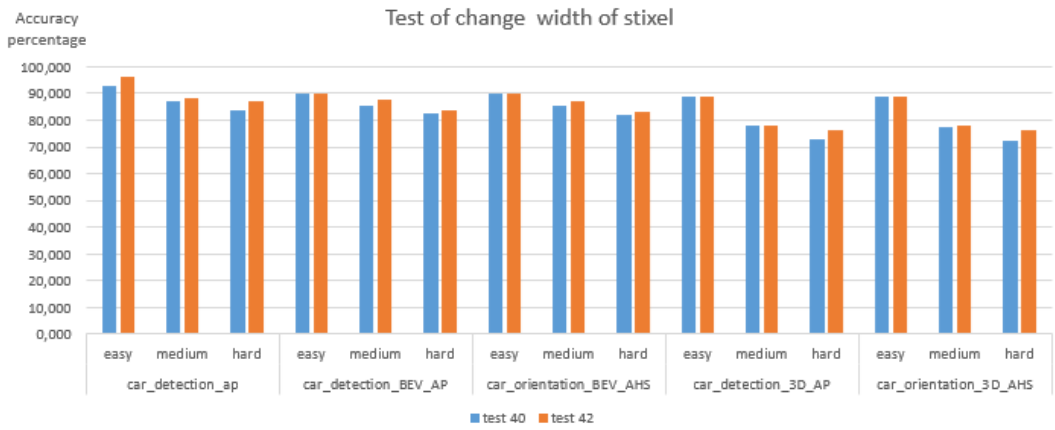


Figure 5.11: Test of change width of stixel

Despite this change, the network does not improve the performance, so it has not to be reported.

5.2.8 Stixel derive from mono e stereo camera

Another group of tests was focus on the stixel obtaining from depth maps generated by the stereo camera and monocular camera. A first trial was to evaluate the network training on point cloud's stixel on stereo's stixels. This has been done in order to test a kind of transfer learning.

The base on this transfer learning is that stixels are a medium type data that summarize a point cloud where each point indicates the distance. It does not care where the point has been extracted because the information of each point

is the same if it has been derived from lidar or a stereo camera. This transfer learning can be very useful because it means that could be possible to train the network with stixel extracted from lidar and in inference use stixel extracted from the camera. This can be useful in order to implement interoperability of the network abstracting this from the sensor used to extrapolate the point cloud and the stixel.

Test 52 is the network trained on test 24 and executed with stixel generate from a stereo camera. Test 53 is the network trained on test 36 and executed with stixel generate from a stereo camera.

Table 5.20: Test training network on stixel from point cloud and test on stixel from stereo

	test 24			test 52			test 53		
metrics	easy	med.	hard	easy	med.	hard	easy	med.	hard
AP	90.56	82.71	78.42	36.77	22.79	19.57	35.77	22.31	18.57
BEV AP	90.24	79.4	76.89	33.22	21.18	18.76	32.81	21.55	18.29
BEV AHS	90.18	79.21	76.52	30.91	19.82	17.63	31.06	20.44	17.38
3D AP	88.60	72.25	67.64	11.24	9.31	9.09	8.27	5.41	5.08
3D AHS	88.55	72.09	67.44	11.18	9.30	9.09	8.13	5.37	5.06

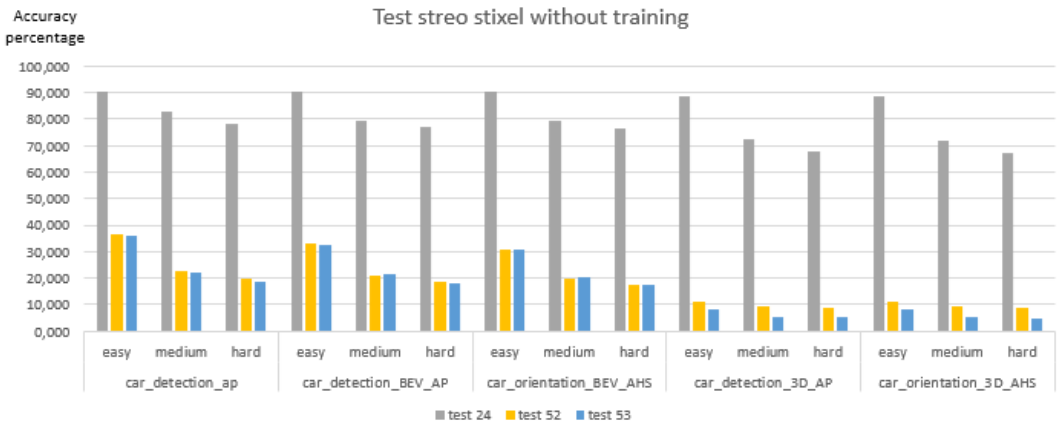


Figure 5.12: Test training network on stixel from point cloud and test on stixel from stereo

As is possible to see in table 5.20 despite the good intent the result does not

satisfy this theory. This could be happening because the depth map generates by stereo and mono camera is less precise than the one generated by the lidar so a network that was training on this kind of data makes a lot of mistakes. Analyzing the result of the inference of the network it can be noticed that the cars that are far from the camera weren't fitted so well from the depth map this decreases the level of accuracy.

This problem can overtake using a more precise network that extracts the depth map.

After this kind of test, it has been decided to training directly the network with stixel generate from stereo and mono camera. In order to exploit the know-how establish from the previous trial for the test, it has been kept the configuration of test 24 and test 36. These configurations have been chosen in order to have a test bench starting with the network that achieves the best result with and without multi-scale grouping.

Table 5.21: Test training on stixel derive from stereo camera

	test 24			test 56			test 57		
metrics	easy	med.	hard	easy	med.	hard	easy	med.	hard
AP	90.56	82.71	78.42	69.05	46.22	39.97	71.78	47.97	44.68
BEV AP	90.24	79.4	76.89	56.40	36.62	31.07	57.99	37.65	32.03
BEV AHS	90.18	79.21	76.52	55.80	36.17	30.66	57.24	36.96	31.41
3D AP	88.60	72.25	67.64	48.02	32.35	27.24	49.01	32.32	26.94
3D AHS	88.55	72.09	67.44	47.57	32.06	26.96	50.07	32.61	26.65

The configuration of the network adopted in test 56 and 62 is the configuration of test 24 and the configuration adopted in test 57 and 63 is the configuration of test 36. Is it possible to see comparing the table 5.20 and table 5.21 training the network directly on stixel generate by stereo camera allow to increase the performance comparing the results the network training with stixel generate on lidar and evaluate on stixel generate by the depth map.

This result underlines that this network can perform on both types of stixel,

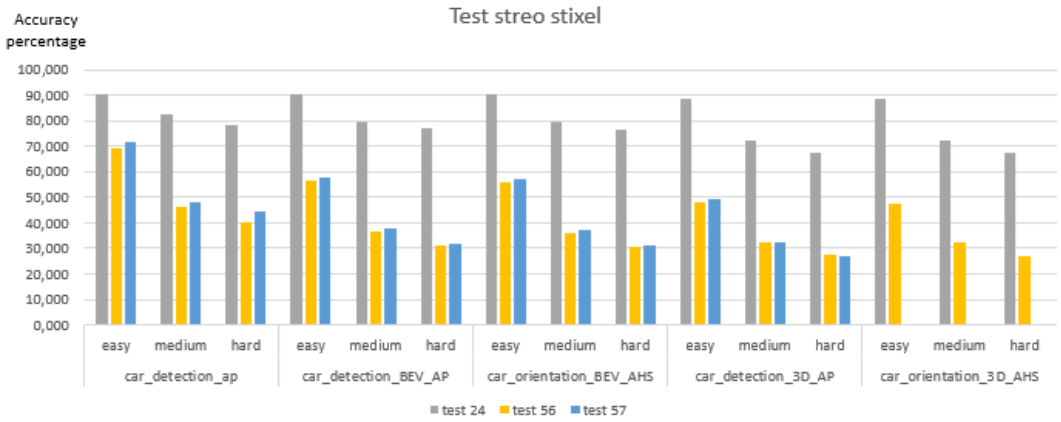


Figure 5.13: Test training on stixel derive from stereo camera

Table 5.22: Test training on stixel derive from mono camera

	test 24			test 62			test 63		
metrics	easy	med.	hard	easy	med.	hard	easy	med.	hard
AP	90.56	82.71	78.42	54.98	37.30	31.83	56.56	38.29	34.06
BEV AP	90.24	79.4	76.89	32.15	23.28	20.86	33.65	23.67	21.21
BEV AHS	90.18	79.21	76.52	31.88	23.08	20.68	33.44	23.49	21.04
3D AP	88.60	72.25	67.64	21.84	15.97	14.85	22.62	16.00	14.67
3D AHS	88.55	72.09	67.44	21.74	15.93	14.81	22.52	15.95	14.63

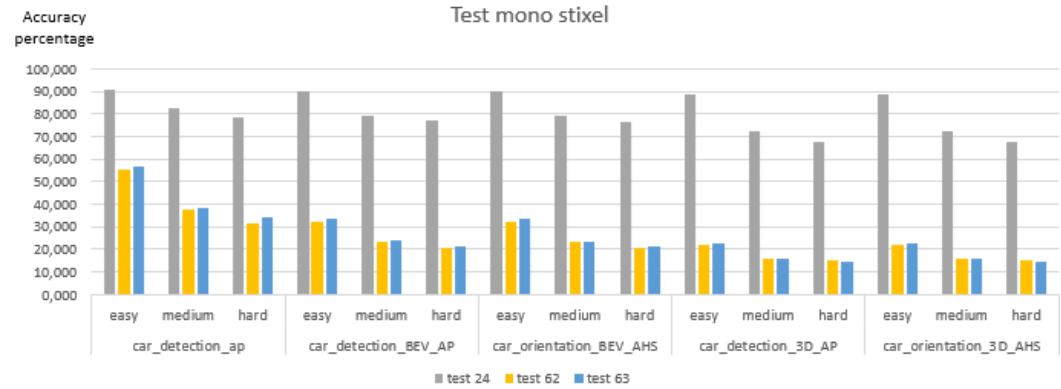


Figure 5.14: Test training on stixel derive from mono camera

but an important part is the accuracy of the starting data. Having rough and imprecise data decrease the performance of the network.

This has been confirmed from the result on stixel derived by the mono camera.

Evaluate a depth map from a mono camera is a very difficult task so nowadays the accuracy of this network is low comparing to that accuracy on a stereo camera or lidar. Due to this lack of accuracy, the results 3d object detection on mono-camera is lower than others.

According to the result on stixel extracted from a lidar better result could be achieved by increasing the accuracy of the network that extract depth map on stereo and mono cameras.

5.3 Memory consumption and inference time

Analyzing the memory consumption of the weights in the tests presented in this thesis is possible to notice that the only change that modifies the memory is the adding of the multiple radii on the SA layer. This is an important result because underline that changing the number of seed points and the type of sampling does not change the memory consumption of the weights.

The weight is fixed a 9,35Mb for the weights of the networks with the SA layer with a single radius, and 13.5Mb for the weights of networks with the SA layer with multiple radii. This simple change increasing on 50% memory consumption and increase the performance by 2%. So in the production phase, it should be avoided in order to have a lighter neural network.

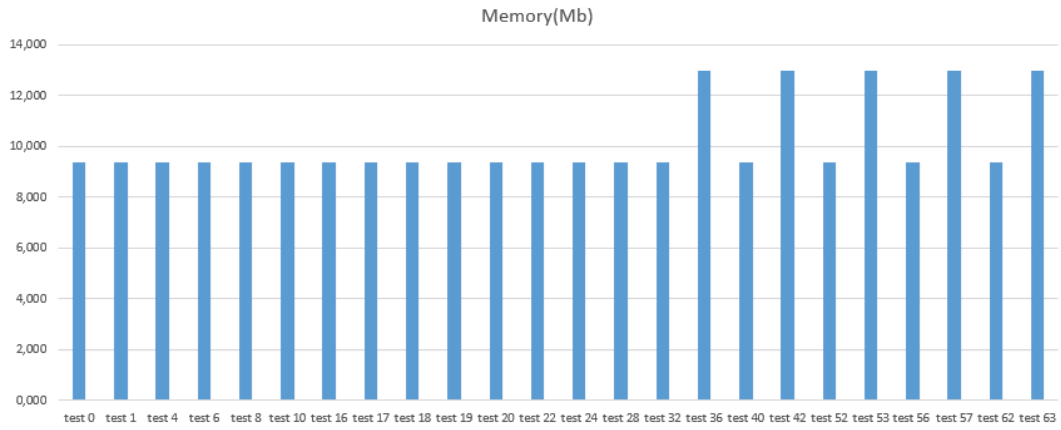


Figure 5.15: Memory consumption of the weights

Another metrics useful for the choice of the set up of the network is the

inference time. The final goal of this network is to achieve a kind of real time computation due to his final usage, autonomous driving. An ideal network should be lighter, faster, and precise. The inference time is linked by the hardware used in this case the GPU where the network has been evaluated is an Nvidia Tesla. The comparison of these metrics is useful in order not to find the absolute inference time but to find the relative inference time regarding the network setup.

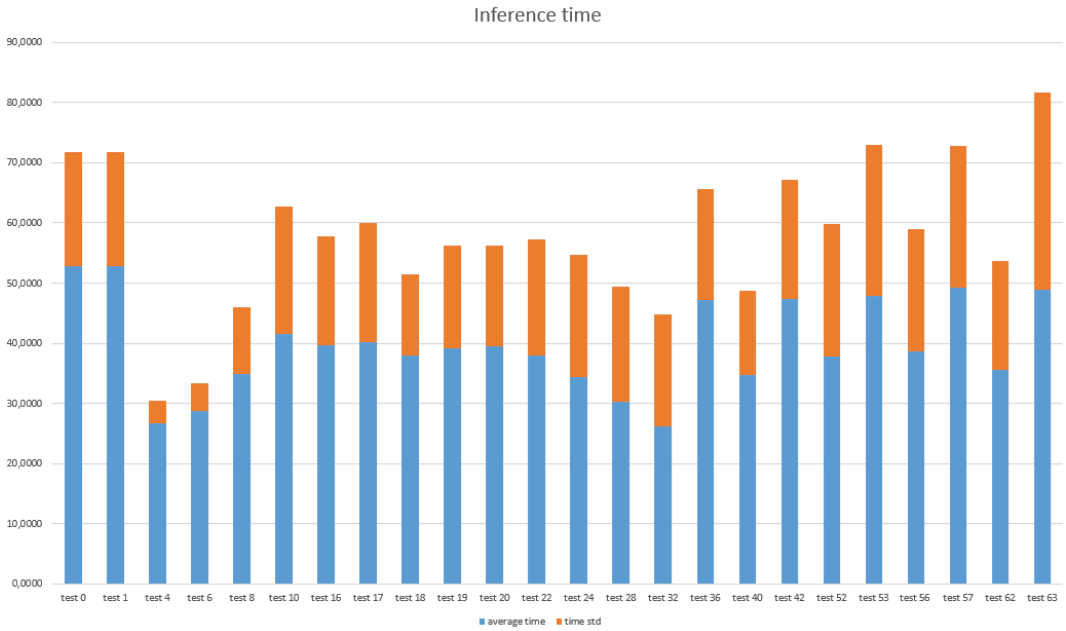


Figure 5.16: Inference time

The figure 5.16 summarize all the inference time obtained in all the tests.

Analyzing them, it can be noticed that using the configuration of stixel with one point instead of the one with four points halves the inference time. This can be derived from the fact that the network has to analyze four times fewer points. This is a relevant result because means with the configuration with one point and allow to improve the result and reduce the inference time keeping the memory usage of the weight constant.

Starting reducing the number of seed points from 4096 to 2048 the network increase the inference time and the standard deviation of this measure. This is not a usual behavior and can be derived from some specific low-level configuration

of CUDA that optimize the network for some specific configuration.

Keeping as a reference point test 10 it is possible to notice that by changing the type of sampling layer the inference time has a small variation so we can figure it as constant.

Reducing the number of seed points there is a reduction of inference time. This can be derived from the fact that each SA layer has to sample and group fewer seed points.

Increasing the number of sampling radius the network increase the inference time. Despite test 36 allow achieving the best result it has both high inference time and memory consumption.

Changing the configuration of input data from stixel generate from stereo camera or stixel generate from mono the inference time not change. This underlines the portability of the network that can ideally achieve the same performance changing the type of input data.

5.4 Comparison with the state of the art

The last part of the result analysis is the comparison with the state of the art. This type of network is quite new taking into account that the oldest paper has less than two years.

The Kitti dataset is split into two subsections the training set and the test set.

The training is also split into two subsections: the training data and the validation data. This split has been officially delivered from the Kitti and it takes into account all the different types of scenes. The training data is used to train the network and the validation data to have a first validation test.

The goal of the test set is to have a standard test bench to verify the result. The ground truth of the test bench is not public so in order to have a result, it has to be submitted on the official Kitti website. This submission has to be approved and it is not so easy. For this thesis, the submission has not been

approved so the network has been evaluated only on validation data. The validation data can be used also for a common test bench but it has not the official approval.

Table 5.23: Performance comparison on KITTI 3D object detection val set for car class

		3D AP car			BEV		
Method	Modality	easy	med.	hard	easy	med.	hard
VoxelNet	L	81.97	65.46	62.85	89.60	84.81	78.57
Point Pillar	L	79.05	74.99	68.30	88.35	86.10	79.83
STD	L	89.70	79.80	79.30	90.50	88.50	88.10
SVGA	L	90.59	80.23	79.15	90.27	89.16	88.11
3DSSD	L	89.71	79.45	78.67			
Frustum	R+L	83.76	70.92	63.65	88.16	84.02	76.44
Test 24	L	88.60	72.25	67.64	90.24	79.4	76.89
Test 36	L	89.05	76.88	72.35	90.39	84.74	79.14

Analyzing the result on validation split on training data the network presented in this thesis achieves high-level results comparing whit the state of the art. The fluctuations on one point are common in the training phase so it can be considered that the network is at the same level comparing with the other networks.

This final result is very important taking into account that the network works on 20 times fewer points. This underlines that having a good compression of rough data and adapt the network to work on that type allows achieving a result that is comparable to the state of the art.

As is possible to see from table 5.23 all the networks achieve at most 90% of accuracy. This can be derived from a bottleneck of the training, generate due to a lack of precision of Kitti dataset annotation. Observing more in the detail the 3D annotation in the images it is possible to see that in some scenes there are cars that are not annotated and in evaluation, phase could be classified as

false positive decreasing the performance of the evaluation.

5.5 Final consideration

In this chapter, it has been presented all the tests and paths take in order to adapt the network to 3D object detection from stixel. The discovery made during this tuning are:

- A smaller batch size equal to 4 allow to improve the performance during the training due to the increase of the ability of the network to generalize;
- The configuration of the stixel composed with one point and width and height as a feature has multiple advantages: it reduces the memory of the dataset, improve the accuracy of the network and reduce the inference time;
- The sampling layer of set abstraction level have a better performance if it samples the seed points taking into account both Euclidean distance and feature distance;
- The use of data augmentation in the training phase increase the performance in a significant way. In particular, the use of global flip, local translation, local rotation, and mixup augmentation increases accuracy. The use of global rotation should be avoided because decrease the performance;
- Adapting and reducing the number of seed point sampled in the SA layer allows to improve the performance adapting the network to the reduced number of data;
- Due to the sparsity of the point cloud, the use of the multi-scale grouping in the Set abstraction level of the backbone increases the performance, but increase also the memory consumption of the weights and the inference time;

- Changing the width of stixel increase the resolution and the number of stixel but not increase the accuracy of the network;
- Training the network with data coming from a more precise sensor like lidar and using them on data coming from a less precise sensor like stereo camera decreases the performance of the network becoming it not reliable. Despite that the network achieves a relevant performance on stixel extrapolate both from the stereo camera, mono camera, and lidar.

This network allows achieving a high-level accuracy despite the starting data were 20 times less than a normal scene. This means that high accuracy is possible to achieve if the starting data are compressed in a good way so compression does not mean reduction of performance if this compression has been made in a smart way.

Figure 5.17: Starting scene

Figure 5.18: Starting points

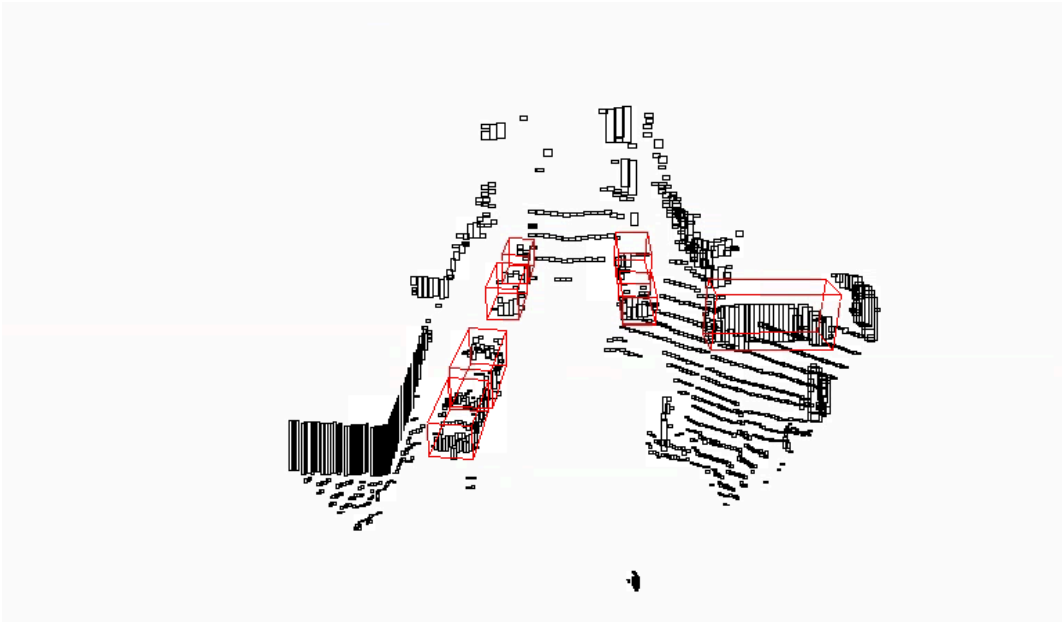


Figure 5.19: Ground truth stixel

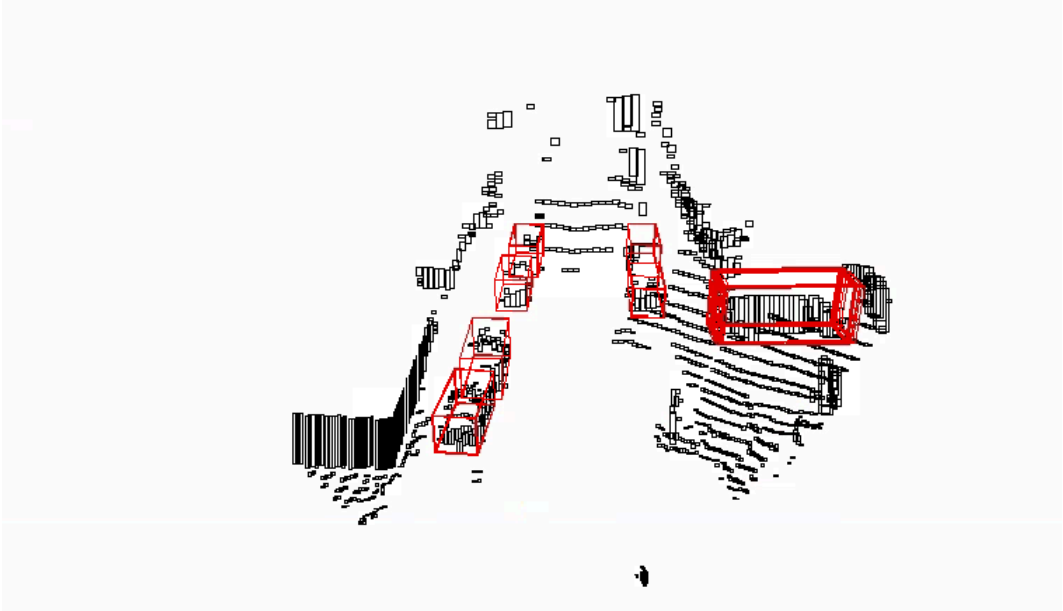


Figure 5.20: Prediction of the network

Chapter 6

Final considerations

This thesis has presented an adaptation of a deep learning algorithm that is able to extract the position 3D and class from a scene in an outdoor environment.

The deep learning algorithm uses as input data a medium type data called stixel. A stixel can be thought like a small rectangle that starts from the base of the road and then it rises until the top of the obstacle summarizing the vertical surface of an object. The algorithm to generate stixel has been developed and implemented in a novel way that has been described in chapter 3. The main pass to create this type of data are:

- the elimination of points that lied on the ground plane;
- the creation of an average matrix that summarizes the depth of group of stixel;
- the creation of stixel merging all the cells that belong on the same object.

The stixel has been created from a point cloud generated by LIDAR, from a depth map generated by stereo and mono camera.

The stixel allows a reduction of points from 40 000 to 1100 from lidar point and from 465 750 to 1000 for depth map points.

The neural network is an adaptation of VoteNet[9] composed of a backbone, a voter layer, and a detector layer.

This network, designed for an indoor environment, has been adapted to work in an outdoor environment having as input medium type data like stixel.

After a documented tuning and designed phase, the network is able to achieve the state of the art of 3D object detection on point cloud on the outdoor environment despite the reduction of 40 times of points. This enforces that it is possible to achieve an excellent result with the right compression of data without losing information.

6.1 Future improvement

The future improvement of this thesis are:

- The creation of stixel starting from a depth map extracted from stereo and mono camera using a more accurate neural network;
- the evaluation of the performance of the network on other datasets like NuScene[25] or private dataset;
- Add other new types of layer like a self-attention layer in order to obtain more accurate discrimination of false positive;
- Implement the network in order to work with real-time data.

Bibliography

[1] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. pages 3354–3361, 2012.

[2] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[3] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. 2017.

[4] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. 2019.

[5] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud, 2019.

[6] Qingdong He, Zhengning Wang, Hao Zeng, Yi Zeng, Shuaicheng Liu, and Bing Zeng. Svga-net: Sparse voxel-graph attention network for 3d object detection from point clouds. 2020.

[7] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. 2017.

[8] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. 2017.

[9] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds. 2019.

[10] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. 2020.

[11] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. 2018.

[12] Hernán Badino, Uwe Franke, and David Pfeiffer. The stixel world - a compact medium level representation of the 3d-world. 5748:51–60, 09 2009.

[13] David Pfeiffer and Uwe Franke. Towards a global optimal multi-layer stixel representation of dense 3d data. pages 51.1–51.12, 2011. http://dx.doi.org/10.5244/C.25.51.

[14] Rodrigo Benenson, Markus Mathias, Radu Timofte, and Luc Van Gool. Fast stixel computation for fast pedestrian detection. pages 11–20, 10 2012.

[15] Lukas Schneider, Marius Cordts, Timo Rehfeld, David Pfeiffer, Markus Enzweiler, Uwe Franke, Marc Pollefeys, and Stefan Roth. Semantic stixels: Depth is not enough. pages 110–117, 06 2016.

[16] Marius Cordts, Timo Rehfeld, Lukas Schneider, David Pfeiffer, Markus Enzweiler, Stefan Roth, Marc Pollefeys, and Uwe Franke. The stixel world: A medium-level representation of traffic scenes. *Image and Vision Computing*, 02 2017.

[17] Daniel Hernandez-Juarez, Lukas Schneider, Antonio Espinosa, David Vázquez, Antonio López, Uwe Franke, Marc Pollefeys, and Juan Moure. Slanted stixels: Representing san francisco's steepest streets. 07 2017.

[18] Florian Piewak, Peter Pinggera, Markus Enzweiler, David Pfeiffer, and J. Zöllner. Improved semantic stixels via multimodal sensor fusion. 09 2018.

[19] Olof Forsberg. Semantic stixels fusing lidar for scene perception. 2018.

[20] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. 2018.

[21] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. 2020.

[22] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77:259–289, 05 2008.

[23] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. 2016.

[24] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. 2017.

[25] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. 2020.