FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Advanced processing and object detection techniques for assisted and autonomous driving systems

Filipa Ramos

U. PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rosaldo Rossetti

Second Supervisor: Alexandre Correia

July 25, 2018

Advanced processing and object detection techniques for assisted and autonomous driving systems

Filipa Ramos

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

July 25, 2018

Abstract

The ultimate dream that has been long standing in the minds of all those concerned or related with the transportation systems is to develop fully operational unmanned vehicles capable of being fully operational by themselves. The rising of autonomous driving could be one of the most decisive technological feats of this era - not only can these systems reduce traffic accidents to a minimal, they will be shockingly more efficient in managing fuel and energy consumption.

On the scope of autonomous driving, the process designated by object detection is one of the basic foundations for self driving vehicles. Being defined as the activity of collecting sensory information which allows the interpretation of all object locations in the vehicle's environment, this process is invaluable on the scope of road navigation. Deep learning has allowed the industry to take a huge step forward in the area. Although, even with all the techniques that have been explored until now, there are still many uncharted areas on the field. For this reason, this work provides an in depth study of methodologies for camera and LiDAR based object detection. The project at hand starts by demonstrating the prowess of the YOLO algorithm both in speed and accuracy through an analysis of its incremental evolution in between versions on the autonomous driving scenario. More than this, a novel network, the HGNet is proposed for segmentation of object areas from an envisioned height grid representation that works through the bird's eye view. Furthermore, a pipeline is described so that information from both sensors can be easily fused and leveraged together in order to obtain three dimensional based spatial and object relationships.

Making use of the renowned KITTI Vision dataset for object detection, the models are severely experimented on and their effectiveness is assessed. Even though the newest version of YOLO seemed to have potential to reach state-of-the-art results, its mean average precision is quite poor on the KITTI object detection benchmark which leads to the conclusion that its applicability on autonomous driving is reduced to a small pool of set conditions. On the other hand, the proposed HGNet is only just a little short of breaking the 80% barrier for f1 score whilst proposing oriented object areas working on incredibly small sized images featuring very few LiDAR sample points at a fast rate.

Keywords: Deep Learning, Computer Vision, Object Detection, Autonomous Driving

Resumo

O sonho transcendente que permanece e floresce nas mentes de todos os indivíduos ou organizações remotamente relacionados com a indústria automóvel é o de desenvolver veículos completamente autónomos que são capazes de operar sem qualquer ajuda externa. O crescimento do apelo à condução autónoma poderá vir a ser um dos feitos tecnológicos mais marcantes desta era - na verdade, estes sistemas não só poderão reduzir os acidentes rodoviários a minímos nunca antes previstos mas também terão um impacto chocante no consumo e gerência do uso dos combustíveis.

Dentro do tópico da condução autónoma, o processo designado como detecção de objetos revela-se como uma das fundações dos veículos não populados. Sendo definido como a atividade de colecionar informação sensorial que permita a interpretação de todas as localizações de objetos no ambiente envolvente ao sistema de transporte, este processo tem um valor incalculável para o cumprimento da navegação nas estradas. O uso de técnicas de *deep learning* tem permitido à indústria uma evolução astronómica na área. No entanto, mesmo com todas as possibilidades já exploradas, ainda existem inúmeras direções por explorar. Devido a isto, este trabalho procura estudar, de forma aprofundada, metodologias para detecão de objectos baseada em cameras e sensores LiDAR. Primeiramente é demonstrado o poder de adaptabilidade do algoritmo YOLO tanto em velocidade como em eficácia através de uma análise incremental da evolução sofrida entre as suas versões no cenário da condução autónoma. Para além disto, uma nova rede neuronal, a HGNet é proposta para fazer segmentação de áreas de objeto a partir de uma representação criada a partir da perspetiva apelidada de bird's eye view. Ainda mais, uma *pipeline* é descrita para permitir uma fusão da informação vinda de ambos sensores de forma a que estes possam ser usados em conjunto para obter relações tridimensionais tanto sobre o espaço como os objetos.

Fazendo ainda uso do reconhecido KITTI Vision dataset para detecção de objetos, os modelos são extensivamente postos à prova e a sua eficácia é avaliada. Apesar do grande potencial que o YOLO demonstrava para obter resultados do estado da arte, a sua precisão média é bastante pobre na benchmark de deteção de objetos do KITTI o que leva à conclusão de que a sua aplicabilidade no cenário da condução autónoma é reduzido para um grupo mínimo de condições predefinidas. Por outro lado, a rede proposta, a HGNet, consegue praticamente atingir eficácias de 80% de f1 score ao propor áreas de objecto orientadas de forma rápida a partir de imagens espantosamente pequenas que contém um grupo reduzido de pontos LiDAR.

Palavras-chave: Aprendizagem Profunda, Visão por Computador, Detecção de Objectos, Condução Autónoma

Acknowledgements

There are so many people without whom I would not have made it here. This goes, sincerely, to all of them. Thank you. In many more ways than you can tell, you have pushed me forward, urged me never to give up my crazily huge ambitions and, first and foremost, told me to believe in myself on the days that even I could not.

Specifically, I give out my appreciation to the whole LiDAR team at Bosch Car Multimedia who received me with open arms and helped me grow not only as a professional but as a person. I feel sincerely grateful towards Alexandre Correia for investing in me and believing in my potential. It was a pleasure to embrace this dreamy project.

For my colleagues at FEUP, I appreciate all the questions answered, even when they seemed rather nonsensical or confusing. I will never forget all the moments we spent growing together, supporting each other through fire and blood. If I am sure of anything is that I could not be finishing this master's degree without the companionship you offered every step of the way. Gustavo Silva, for all the precious time you did not have and still dedicated to helping me scientifically and personally, thank you. I would certainly not be able to be proud of this work the way that I am right now without you. I believed I could do it because of you.

From my own faculty as well, my supervisor, Rosaldo Rossetti, who always had the most supportive position, who helped me even through holidays, weekends and let me decide where I wanted to go with this work every step of the way. I feel very thankful for your mentoring.

For the family that I did not choose but found, my friends Rita Carvalho, Sara Alves and Manuela Rodrigues. Even through the distance, you were always my beacon of light. Your support means the world to me and has kept me going through the darkest of times. For all the conversations, for all the fights and for all the hugs, thank you. I will never forget what you have done for me.

Finally, I give a special mention to my family. You, more than anyone else, suffered through everything with me. You, more than anyone else, know that it was not easy to get here. If I am able to be persistent and not to give up in the face of difficulty it is because you always showed me so through your own resilience. My mother and father, I can only say you inspire me every day and I only wish that I could grow to be such amazing people as you are.

Filipa Ramos

“After all, the ultimate goal of all research is not objectivity, but truth.”

Helene Deutsch

“Faithless is he that says farewell when the road darkens.”

J.R.R. Tolkien

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

2D	2 Dimensional
3D	3 Dimensional
BEV	Bird’s Eye View
CE	Cross Entropy
CNN	Convolutional Neural Network
FCNN	Fully Convolutional Neural Network
FEUP	*Faculdade de Engenharia da Universidade do Porto*
FFNN	Feed Forward Neural Network
FL	Focal Loss
FOV	Field Of View
FPN	Feature Pyramid Network
FR-CNN	Fast Region Based Convolutional Neural Network
HGNet	Height Grid Network
IoU	Intersection of Union
LiDAR	Light Detection And Ranging
mAP	Mean Average Precision
R-CNN	Region Based Convolutional Neural Network
YOLO	You Only Look Once
YOLO9000	You Only Look Once 9000 Class Prediction

Chapter 1

Introduction

It has been a long lasting dream throughout generations of scientists and engineers dedicated to the transportation market to develop technology that would allow vehicles to confidently navigate all kinds of roads in an independent manner. In order to make this dream come true, many teams filled with talented people from all around the world have gathered to study and perfect such a mythic technology. However, even with so much investment, there are still many challenges that have yet to be solved. The setting of autonomous driving poses as a natural hurdle since the current machines do not have yet the sense to compute many types of operations that humans do naturally and the sensibility to make decisions with the necessary trustable degree.

Autonomous driving is described as the process of navigation of unmanned vehicles through the detection of surroundings that allow the system to make intelligent decisions. The first step in the automation of vehicles is then giving computers the ability to constantly understand the environment around itself in the most accurate and speedy way possible. In fact, one of the first and biggest challenges of autonomous driving is the recognition of objects around the vehicle so as to allow it to make the decisions necessary to take passengers in between their destinations. Although many studies have become successful in achieving this task with a somewhat satisfactory degree of confidence, it was not until the development and discovery of neural networks and deep learning that the task of object recognition gained its momentum towards the integration in real world vehicles. Framed in this setting, this work then focuses on the exploration of existent and new techniques for object detection or operations supporting it. In order to ascertain how this science can become a decisive factor in the steps towards the automated driving industry's dream, all methodologies that were created or worked with are thoroughly experimented on in order to better predict and understand their behaviour in possible world like situations.

Aside from the core issue at hand, no matter the methods used in the autonomous driving environment recognition, there is no approach that does not rely on the use of hardware support in order to gather the sensory information from the vehicle's outward system. On this topic as well, many sensors in a wide range of areas have been made use of in order to support the intricate task.

Through the mixed use of RGB cameras and LiDAR, object detection methodologies are explored both in the sense of combining both sensory input data or single-handedly leveraging either. A comparison of their own intrinsic shortcomings is ensued in order to ensure a correct inference on the models' accuracy. Furthermore, it is also interesting to observe what types of information can be extracted for each object on each representation.

In order to achieve a working flow for an autonomous vehicle a pipeline for fusion of detection outputs on camera and LiDAR is proposed. Specifically on this pipeline, implementation is focused on each individual path. On the topic of RGB camera based detection, the YOLO model is assessed and explored on the KITTI Vision Benchmark through the testing of many fold configurations and hyper parameters. Results are then thoroughly reported. Furthermore, on the LiDAR based detection, a network is proposed for segmentation of object areas in frustum.

This work aims to unfold itself as an exploratory project. Testing new approaches is not always completely successful and for progress to happen many others have to fail and ascertain which paths must be closed and which paths we must press on through. On this sense, the implementation relies on the report of a bundle of experiments, whether they showcased proficient results or not so as to underline topics that would be promising in future research efforts. There are still many unknown variables and intricacies on the process of object detection. Exploring the unknown may bring new perspectives to the table, inspiring future works that gain a new direction and bring out great results.

Through out the present work, one can sense a natural flow that starts with an overview of the techniques that are being used for object detection on the current research works. Following the state-of-the-art, the theoretical foundations are described, providing an overview of the specific models and deep learning techniques that will be discussed later so as to provide the reader with a succinct notion of the concepts in play. Chapter 3 reflects on the methods of operation, shortcomings and applications for the sensors in question. The objective is to bring forward the concept of sensory perception. The referred chapter also presents a more theoretical and driven problem formulation on object detection and their constraints. Implementation remarks are then ensued, starting from the pipeline towards the explanation of the tested experiments and their scopes of operation. Finally, all conclusions are drawn, the future paths for research are delineated and the document comes to its closing.

1.1 Context

The present work was developed at Bosch Car Multimedia S.A., rightly framed in their starting project of advancement of processing and object identification techniques for autonomous and assisted driving systems.

Bosch Car Multimedia S.A. is a filial branch of Group Bosch which has a huge presence in several wide ranging markets such as home appliances, security systems and mobility solutions. The mission of Bosch Car Multimedia S.A. is to provide intelligent solutions in areas integrated in driving systems as broad as entertainment, communication and driving assistance.

Bosch Car Multimedia has always aimed at shaping the vehicles of tomorrow and this project comes associated with the ever standing wish to ensure safety on the road. Bringing the perceptive vehicle to life would be the greatest step forward for the automobile industry.

1.2 Motivation and Objectives

Autonomous driving is, as mentioned before, a huge dream that humanity wishes to achieve in the future. However, the issue is rather problematic, intricate and complex and this is reflected by the many challenges still faced by scientists and engineers in the area alike. One of the biggest monsters in the way is the fact that every method applied on these systems needs to be absolutely fault proof. Critical settings such as this one require a yet to reach level of confidence on computations so as to guarantee the safety of the passengers at all times. This means that not only do these methodologies for autonomous driving need to be 100% accurate, they need to be faster than real life threatening situations. Characteristics such as portability, lightness and performance have as much importance as the results outputted by the software.

Even more challenging than the technological and scientific angles of autonomous driving remains the work to be done on a social level. No technology can be successful may it not be accepted and understood by society. A critical system such as a self driving car is still a step too grand in the minds of many and its possibility of failure remains one shadow overbearing the confidence that possible users may express towards the system.

As far as autonomous driving goes, the impacts that such an innovation can bring to society are far reaching and their relevance speaks for itself. First of all, these types of vehicles can help reduce accidents and decrease traffic which leads to safer roads for both pedestrians and other vehicles alike. Moreover, these systems can have a great impact on the fuel and energetic consumption, bringing sustainability one step further for these transportation systems. Besides this, autonomous vehicles can be a decisive factor to improve the life quality of motor impaired drivers which have high difficulties in driving by themselves. On a general plane, advancing autonomous driving can definitively have a major impact on people's lives and thus these impacts are the first motivators for the present work.

On another plane, there is also value to be found on the fact that this project can provide a platform for future exploration of the use of deep learning in the several possible settings of the phases of object recognition. It aims to be a step towards filling the void still present on the issue and to show both researchers and clients alike that a self sufficient driving system can be a reality without fear or consequence.

Specifically on the topic of object detection, the applications of a reliable software capable of understanding its environment is much larger than that of autonomous driving. Advancements on this field can bring about incredible potential products that range from daily life uses to complicated security systems. In general, object detection is motivated by its many unfolding applications and by the impact it could possibly have on the lives of the future.

The main line of objectives is lead by the exploration of techniques for object identification both in camera and LiDAR sensor settings as well as the combination of information obtained from both. Furthermore, this work pretends to be a platform for more research in the sense of exploration of new methodologies and perspectives that may lead to the finding of complete, final solutions to these issues.

1.3 Document Structure

The remaining of this work comprises 5 more chapters. Their explicit content is summarized below.

Chapter 2 The state of the art in object detection techniques is explored both on the two and three dimensional planes. Some related works in the area of both general purpose and autonomous driving specific methodologies for object finding are thoroughly explained. Theoretical foundations are laid out so that the reader is informed on the models that were put to use or even inspired some implementation paths that are described on the following chapters.

Chapter 3 Sensory perception is explained alongside with the setup of the platform for data collection employed by KITTI's team. RGB camera and LiDAR sensors follow, shining light on their methods of operation and a discussion of their applicability on autonomous driving scenarios. Furthermore, a comparison is ensued which intends to clarify each sensor's strengths and weaknesses. The issue of object detection is then thoroughly explored in the sense of autonomous driving through the found constraints, obstacles and some evaluation metrics to testify for the methods accuracy.

Chapter 4 Some statistics on the datasets are presented. A pipeline is built for joining camera and LiDAR based information that would be extracted separately. It is shown how these sensors, together, can lead to the regression of 3D bounding boxes. The YOLO model is experimented on, following an incremental process that intends to uncover its applicability value on the issue at hand. The end results are then discussed on the benchmark environment so as to provide a comparison point with literature.

Chapter 5 LiDAR based object detection experimentation is described. A search space reduction technique called frustum is adapted from [QLW+17]. A representation based on bird's eye view elevation images is then laid out. Furthermore, a network, the HGNet, is proposed for LiDAR object area segmentation on this height grid representation. Starting from its architecture, data augmentation processes and visualized extractions, some experiments are carried out in order to better understand the network's effectiveness. Besides this, a loss function based off of the focal loss is explored.

Conclusion 6 A summary of all the work serves the purpose of enlightening the reader on what was built. A critical analysis of the methodologies employed is then carried out, showcasing both the strengths and weaknesses of the work achieved. Building up from what was found to be lacking, directions for improvement are identified. Purposely following that, new paths for possible research are also delineated.

Chapter 2

Deep Learning and the Problematic of Object Detection

The extraction of object shapes and identification from several types of input data such as camera images, laser and LiDAR point clouds is an issue that has gained its relevance and momentum from the turn of the century. Older methodologies that surfaced when the issue of object detection started to gain relevance relied on the possibility of recognition of object's geometrical features. Progressive thinking believes that there is much more to extract than the standard geometry intricacies when doing object recognition. On this topic, it is stated in recent works that objects can have an hierarchical relationship, for example, some objects may be placed inside, on top or under other objects. Extracting these relationships becomes then important in order to completely have machines understand the involving environment. Furthermore, a whole object can have in itself a panoply of colors, textures and combination of shapes which complicate the process even more since discerning in between these characteristics does not always signify discerning objects.

As one can see, the issue of object detection is far more complex and wide in application than it seems at first sight. For this reason, it is important to dive into the work that has been carried out so far in the area. Understanding what is behind us is the key to creating relevant innovation.

2.1 Automated Object Detection

The issue at hand - object detection - has been a constant topic addressed in literature due to its complexity and still current lack of reliability at the level necessary for autonomous driving. The methodologies adopted all over the literature in early years seemed to shift towards the mentality that 2 dimensional data is easier to process. Hereby, many algorithms surfaced on the area of image processing and recognition. At the same time, the recognition of objects in LiDAR was also carried out through 2D based representations, such as [QCS+16], being more often than not abstracted to 2 dimensional settings. An example of this behaviour can be found in [Mel15] in which the author first projects the 3D data into a 2D setting, uses a region based convolutional neural network to detect objects and exports the results back to a 3D environment.

Overall, for both sensors, the approaches taken can be segmented into two major groups: data segmentation with geometrical based extracted features or unsupervised learning of the features to observe from the data itself.

The first type of techniques hope to extract by hand sort of geometrically based features that represent the objects. They rely either on the recognition of line segments, circles or other basic kinds of shapes. All the approaches that fall on this type have the basis that in order to separate the objects on the scene, their basic forms have to be considered.

Even with all the advancement, even if camera exploration has been done much more widely, object detection on either of them is not yet considered as solved. Specially in a three dimensional setting, the interpretation and use of effective methods is still poorly explored. The fact that these models need to be fast and accurate all at the same time contributes to the difficulty of the achievement of a whole solution.

Since understanding what has already been done is the key to producing newer, effective methodologies, the work on the field is summarized through the following sections.

2.1.1 Camera Detection

As a general issue, camera based object detection has suffered many advancements already and is currently enjoying its greatest times as the accuracy of the methods discovered hits the roof. Since much work is available on the area, the divisions below aim at giving a general idea of what researchers are applying on this field to achieve such great accuracy levels.

Starting from primal algorithms that extracted features by hand from images, the field of camera based detection has come a long way. In fact, in the beginning, the methodologies struggled to find the best features to extract manually, having a bundle of authors released many different views on the subject. Whilst some believed that shapes were the most important features, others relied on textures and color if available. However, it was fast understood that objects might be occluded and thus shapes can be twisted or destroyed in the uniform sense. More than this, texture often occurs differently on several places of the object alongside with color being absolutely variable even on standard, equal objects. These hand by hand extractions were not enough and the field did not get a boost until the real applications of convolutional neural networks were discovered.

At the start of it all was the AlexNet [KSH12], a deep convolutional neural network that showcased impressive results on the ImageNet Large-Scale Visual Recognition Challenge never seen before with the use of CNN. This starting architecture was simple in design, featuring simply 5 convolutional layers alongside with 3 fully connected layers. However, it was the start of the reign of the deep convolutional networks on the field. Inspired by Alex Krizhevsky et al.'s efforts, the millions of works that would come after brought incredible progress, leaving AlexNet's mark on the field forever.

From the bundle of projects that would come forward, there are some that have played a more relevant part on the issue at hand. For example, the ZF Net [ZF14], which came out shortly after, showcased how to optimize AlexNet even further towards higher accuracy performance. Besides, it had a great impact on the way convolutional networks were perceived and analyzed. Another important piece of the contributions made by Zeiler et al. [ZF14] was the introduction of deconvolution. A deconvolution is usually described as a path back to the image pixels and was mostly used in order to understand which images created which feature maps.

Another milestone on the computer vision scenario was the VGG Net [SZ14]. The introduction that most shocked the community on this paper was the combination of great results with a general simplicity on the convolutions applied. In fact, the architecture of this network always made use of $3x3$ filters with stride 1 alongside with $2x2$ max pooling layers with stride 2. Even though the use of these sizes for filters and strides are common currently, before VGG Net they were seldom recurred to. And yet, nowadays, models like YOLO [RFA18] still make use of these simple concepts.

Moving even more forward, building up from AlexNet and what followed, deep learning was established as the way to go for object detection. Influenced by so, the evolution on the field of generally referred to as two dimensional object detection has been characterized by two main different approaches. These methodologies seem to divide themselves into two stage detectors, which seem to prime in accuracy and the one stage detectors which are usually the fastest.

The concept of two stage detectors really kicked off with the surface of the R-CNN [GDDM14] model. What this model had in accuracy, it did not have in detection speed which invalidated it for such scenarios as autonomous transportation systems. In fact, the original system took over 40 seconds to perform detection on a single image. Over the years, many updates were proposed namely the Fast R-CNN [Gir15], the Faster R-CNN [RHGS17] and most currently the Mask R-CNN [HGDG17] which specializes in object instance segmentation. Lenc et al. even proposed R-CNN minus r [LV15] which used static bounding box proposals instead of the selective search [UVGS13]. Furthermore, through further improvements made on R-CNN, specifically from Faster R-CNN [RHGS17], the selective search [UVGS13] ended up being substituted for a fully qualified region proposal neural network. Selective search proposals were estimated to take around 2 seconds per image which invalidates its application on a real time scenario. Incrementally, the updates that the model suffered unified the network, reducing the strenuous process of having to train or tune independently the feature extraction convolutional neural network, the box scoring SVM, the bounding box adjustment linear model and the non-max suppression step. Unifying this pipeline meant the network became much faster.

At its core, the R-CNN [GDDM14] intends to divide the process of object finding and classification into two different pipelines. The first pipeline, as seen from steps 1-2 on figure 2.1, is focused on proposing regions of interest on which is probable to find objects. This starting process has suffered many increments over the years as it has been previously mentioned. Following that, on the regions of interest, CNN are applied in order to extract object related features, discern whether there is truly something recognizable there and then classification into a group is made. Like all object detectors, the R-CNN includes previous training on the ImageNet [RDS+15]

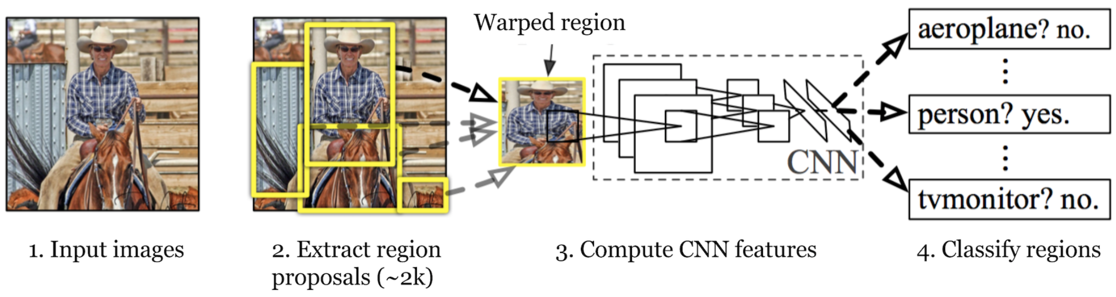


Figure 2.1: From the *Rich feature hierarchies for accurate object detection and semantic segmentation* [GDDM14] paper, R-CNN detection pipeline.

dataset. The ImageNet dataset was truly a remarkable advancement for all flat image based processing tasks like object detection and much more similar or related fields.

In order to provide a more in depth insight into the R-CNN and due to its importance to the field, the following paragraphs will dwell on its internal workings through the updates.

R-CNN uses a region proposal network to achieve the already referred to as the first step. The region proposal network works with sliding windows that run spatially on the features extracted through convolution. Using this process, anchor boxes are built with varied sizes. Each anchor is then sorted with a value which is the output of the following formula:

$$p* = \begin{cases} 1, & IoU > 0.7 \\ -1, & IoU < 0.3 \\ 0, & otherwise \end{cases} \tag{2.1}$$

The value of IoU represents the relation between the anchors proposed and the true boxes and is calculated through the following equation:

$$IoU = \frac{anchor \cap box}{anchor \cup box} \tag{2.2}$$

Using then a regression tool, the coordinates for the regions are proposed using the $p*$ values that were obtained on each of the anchors.

Since selective search could propose the most probable region locations, this output was then parametrized and flattened in order to be fed to a convolutional neural network that extracts the prominent features for the regions. The feature vectors serve as an input to SVM classifiers which discern the object type and labeling. Besides this, the R-CNN proposed in [GDDM14] also considers the effectiveness of drawing a box per object and improve this process through the feeding of the feature vector to a bounding box regressor that obtains the most accurate coordinates to bound the object in question.

As the final step, redundancy is considered and avoided through the identification and suppression of bounding boxes that overlap by a certain desired tunable limit.

As it has been previously stated, this architecture revealed outstanding results in accuracy. However, the running time reported to process, identify and classify all objects within a single image was 53 seconds which is evidently impossible to apply in an autonomous driving context. Furthermore, the model was so complex that it required expensive computations per image which is another make or break factor in a vehicle in movement. Due to these reasons, several researchers created variations of the algorithm proposed by Girshick *et al.* in order to tackle these specific issues.

The first variation that would be impactful was named Fast R-CNN and it was a follow-up work by Girshick (one of the authors of R-CNN), presented on the paper [Gir15]. The main topic of improvement proposed was to share the convolutional layers to extract features for more than one region at the same time. Moreover, it was declared that the region proposals could be done after the features had already been extracted through convolution. This enabled the architecture to have far better performance.

However, there was even more room for improvement and so several authors worked on improving this type of network even more. This movement lead to the birth of the Faster R-CNNN [RHGS17]. This network aimed to be faster and less computational expensive with only one small detailed change from Fast R-CNN. On this approach, only the last feature maps extracted from convolution were selectively searched for region boundaries. This effectively reduced the necessary computations for selective search.

The work by Ren *et al.* [RHGS17] relies heavily on a region proposal network that selects object locations from the image. They also implement a sharing system between full-image convolutional features and the detection network, which enables them to reach an almost cost-free region proposal methodology. The R-CNN uses then these region proposals in order to detect the objects in a straightforward method following up on what has been described previously.

Further than the great achievements in the field of object detection that the R-CNN brought, its launch created a raised interest on unifying network's pipelines under one compiled and more simple training architecture. This was the sense of development followed through Fast R-CNN and Faster R-CNN likewise. As well on the topic of unified architectures, the YOLO9000 paper [RF17] even proposes a method for training a joint model for detection and classification simultaneously. Even though these are not directly object detection related principles, they are methodologies that can bring great effect when applied in those settings.

On the topic of one stage detection, the field was dominated by models such as SSD [LSZ$^+$16] and YOLO [RFA18] for quite some time. The SSD, Single Shot MultiBox Detector, relied on loosing the feature re-sampling step and the evaluation of default boxes for feature maps of different sizes. This model surpassed, at the time, its competitor YOLO version 1. Following SSD, Fu et al proposed DSSD [FLR$^+$17] which introduced feed forward modules with deconvolutional layers which improved accuracy over the original model. However, the contribution did not seem to be significant in terms of detection speed, having approximately equal and at some resolutions even worse detection times than its predecessor.

Recently as well, Facebook's FAIR proposed a novel network, the Retinanet which has been performing outstandingly well in both accuracy and speed. This network is inserted in that of 1 stage detectors, however, it uses a new balanced focal loss which reduces the impact of imbalance between foreground and background classes on the data. The Retinanet detector's backbone is based on the feature pyramid network [LDG+17] which has established itself as a strong architecture choice on the field of deep learning for object detection and convolutional related tasks. This network uses an hierarchy of pyramidal features each merged and combined following a top-down order. The existence of both lateral and top down connections in between feature maps enables the network to be more accurate and fast at the same time.

2.1.2 Spatial Representations

The task of object recognition from point cloud data is a complex operation that has puzzled researchers over the years. From the get to go, several challenges need to be considered on terms of the collected data. In fact, LiDAR can become corrupted and confusing to interpret when objects are overlapping from the perspective of the sensor. Furthermore, due to the beam reflection based approach that this particular sensor uses, objects hidden behind other objects become impossible to see or discern in the point cloud data. Many occurrences may lead to noisy datasets with corrupted geometric features. The authors of [BSC12] have reflected upon this issue, pointing the importance of selection and extraction of meaningful features from LiDAR data. They propose several histogram descriptors that reflect the distribution of the points in order to guarantee a successful analysis of the data. Several works have dangled on the possibility of fitting data with statistical distributions and have been successful in doing so in a wide range of application areas.

Even more, some more issues might be spotted upon the operation of collecting the data. As has been stated by several authors before, in particular in [MNG17], if data is collected in real time, inconsistency may start to show in between frames. The authors also mention the difficulty of analysis of geometrical shapes for when the sensor is in motion, the objects appear distorted due to the changing position of the vehicle. Furthermore, when analyzing LiDAR data collected in real time, some frames may have the same object represented in different angles. Instead of classifying and recognizing this object again, it is more efficient to fuse this kind of data. Based on these facts, the paper first presents a motion correction technique in order to cope with displacement errors. This methodology suggests computing the timestamp of the initialization of each channel's laser beam output and using the resolution and duration of the scan to work out the transformation matrix that will then be used to correct the point cloud measurements. Furthermore, a temporal fusion in between frames which show the same objects in different positions and distortions is proposed in order to make the point cloud more dense and allow for a more easy object filtering operation.

Taking in mind such considerations, the procedure of processing LiDAR data into usable representations is then an important step before the start of object detection concerns.

Before deep learning was revealed, the identification of objects was made in the most varied approaches. Overall, the methods found in literature tried to represent the data in structures more easily handled and in which objects could become more evident. These methods can be grouped mainly into parametric, clustering, occupancy grid and break points detection. Even though these groups enclose the most widely known and applied techniques, several other approaches that understand geometrical considerations have also been explored.

Parametric methods consist of the identification of line segments or any kind of geometrical shape such as circles or ellipses that can be fit to the points in analysis. Approaches such as this tend to have a great performance in indoors environments since the objects are more geometrically defined. However, in the outdoor setting, it becomes harder to discern shapes specially in objects such as trees, bushes and all kind of plants. Some of the most well known methods that can be fit into the parametric approach are the RANSAC (as presented by Martin A. Fischler and Robert C. Bolles in [FB81]), split and merge (for example [TRC$^+$]) and iterative end point fit (as in [MYLX11]).

RANSAC has been a very impactful algorithm in a wide range of applications. As was presented at SRI International[1] by Fischler *et al.*, this method allows the estimation of the best model parameters in an iterative way that has the higher probability of succeeding as the number of iterations increases. The algorithm was first used in order to find a solution to the Location Determination Problem (LDP). The basic assumption of this approach is that inlier points can be explained by certain parameters that follow a distribution. For this reason, the identification of the inliers allowed RANSAC to be effective at recognizing object shapes such as lines and circles. Since outliers were identified, the noise could also be removed from the input data.

Once researchers realized the potential of this techniques on other areas, it was applied to more object detection related environments. As such, it was one of the first methodologies that had an interesting approach at the issue.

Besides RANSAC, the split and merge approach was also proposed in order to segment scenes and eventually to solve the issue of object recognition. This technique used in many works such as [TRC$^+$], [WT04] and [MYLX11] consists of dividing the scene into uniform parts based on homogeneity. If the region is found not to be homogeneous, then it is split once more until the condition is reached. Finally, similar regions are merged back together. This made it possible to discern, specially in images, areas that presented similar traits and might have represented objects in itself. However, having an homogeneous region does not necessarily mean that it is meaningful as an object representation and since data can show different features all in the same object's area, this algorithm was best fitted to binary data and not so much sensor information obtained from vehicles even though it has been widely applied in such settings. For example, the work by Teo *et al.* [TRC$^+$] uses split and merge as a tool to process LiDAR data. On this work, split and merge is used specifically to identify the shapes of buildings.

On the other hand, clustering methodologies rely on the principle that the data points can be grouped together and represent a certain object. This way, clustering algorithms are applied in order to grasp the objects on the scene. However, as mentioned in [PJ17], clustering is also a complex process which implicates the use of a function that expresses correctly distance in between points and clusters. This is a known hard issue of clustering. Finding metric functions that

[1]SRI International is a research institute located in California that belongs to the Stanford University.

express the real relationships of data in the real world and sometimes finding distance preserving embeddings of these is not a trivial problem. In fact, sometimes, such functions do not exist at all. Moreover, the constant queries needed for distance comparison can become an heavily expensive operation which can not be fit for use in vehicles or applied in the contexts of autonomous driving.

Related to these methods, the authors of [KWB08] propose the use of not only the distance but the angular disposition as well when building clusters. The work also presents a counter evasive rule to avoid the high number of comparison queries. The use of the kd-trees structure also makes the process more efficient as they are proved to be specialized in queries that are radially bounded. This work reflects on the two types of clustering possible for this issue - hierarchy and partition. Using the hierarchy method of clustering means finding levels of similarity in between objects which are reflected using a dendogram. However, this approach is disregarded since in laser data the relationships between objects are often irrelevant. For this reason, the proposed radially bounded clustering focuses on partitioning the set of points.

On the topic of occupancy grids, there is a wide range of varied algorithms which have proved their worth in many different applications of 3D data interpretation. The original concept is to project 3D data points unto a 2 dimensional grid identifying cells with the parameters obtained from the sensor (for example height). A connected components algorithm is then applied to identify the objects. This specific methodology using occupancy grids was introduced by Rubio *et al.* in [RLR13]. However, connected components have also been used single handedly in order to segment LiDAR data. Even though Rubio *et al.* present a novel method applicable to 2D LiDAR data [RLR13], they nevertheless achieved scene segmentation using a variation of the algorithm generally called one-pass connected components.

Much like the previous approaches, there are some limitations and dangers in the application of such algorithms. On LiDAR data, the furthest the object is the less dense is its point cloud. This is a threatening setting to the occupancy grids since it may lead to overfitting for points not connected may belong to the same object in reality and be segmented as different items through the algorithm. On the contrary, the closer the objects are to the sensor the more dense will be the final point clouds which leads to an underfitting plane with several different objects being identified as one item only. The adjustment of the size of the grid has a huge impact on its performance and is one of the biggest hurdles to overcome when implementing such an approach. This is the main reason why structures such as the quadtrees were created. Langerwisch *et al.* have even proposed a new quadtree structure to support a new inverse laser scanner sensor in their work [LW13].

Techniques such as these have been widely implemented in the literature. Specifically in the autonomous industry, a variation called *Pyramid Grid Processing* is introduced in [RLW$^+$09] included in a system that predicts the size, direction and speed of the movement of objects from LiDAR data. This grid introduces the tracking of object velocity besides the standard occupation/free space cell information holding.

Besides this, another variation presented by Weiss *et al.* on the work [WSD07] proposed using a grid of probabilities which were updated at each real time scan of the LiDAR sensor using the Bayes Rule. These probabilities model the possibility of the cell being occupied or being free on the real data.

Occupancy grids can also be an invaluable asset as a preprocessing step before segmentation of the point cloud into objects. This goes even for deep learning approaches. Some works explore some segmentation with grids in order to remove the ground and back planes. An example of this type of preprocessing can be found on the work of Börcs *et al.* [BNB17] in which LiDAR data is processed.

Techniques associated with break points segmentation aim to discern different objects through the detection of the spaces between the connections inside the data. Some authors have proposed methods based on this principle as is the example of Zhang *et al.* in [ZATX03] who based their implementation on the use of an Extended Kalman Filter. Further works on the principle estimate the boundaries of the objects.

The authors of [MNG17] discuss another range of techniques - the tree based - as opposed to the grid based which the occupancy grids, parametric methods and voxels are an example of. On the mentioned work, two groups of segmentation based techniques are reported as the octree and range tree.

Following the mention of voxels, the works [APPN16] and [BP16] are also an example of its use in object detection. Moreover, mentioned below in more depth, voxelization is widely used as an input data representation for neural networks.

Wang *et al.* reflects on the possibility of using structures such as octrees in order to successfully interpret LiDAR 3D data. Specifying on the possibility to grasp spacial features hidden in the depth dimension of the third dimensionality, their work [WT04] presents a proposal for a variant split and merge algorithm which is based off uniquely on the octree. The pipeline of their achievements describes a starting point with raw unprocessed LiDAR data becoming an output in 3D planes filled with conceptual information from the scene.

Even if these are the most widely known and popular methodologies that surfaced on the start of the century, several researchers also appointed a variety of other possible approaches to the issue in question. On this area, the work of Magnier *et al.* in [MGG17] presents an unseen method which relies on the belief theory to merge criteria of the several features and predict the objects in the scene using a voting strategy. On this proposed algorithm even classification is carried on using the belief system and the fusion of all the collected beliefs with their confidence degree which are processed using a Kalman Filter.

On the use of varied techniques, [HL09] achieve a real time classification system making use of point feature histograms. On their approach, point cloud is segmented in two and a half occupancy grids. This is also an example of the use of occupancy grids as a processing tool.

2.1.3 LiDAR Detection

Even if these image based methodologies have a long history and a curriculum of successes in various applications, as 3D dimensional data started to be easily accessible with various types of sensors within vehicles, the focus of research shifted towards the three dimensional space, even if keeping the inspiration from the discoveries made on the 2D field.

An example of such adaptations and based mostly off of the R-CNN, some authors adjusted that same network towards a more three dimensional oriented setting. Based on this concept, a Faster R-CNN was employed to process LiDAR data and recognize objects in the work of TJ Melanson [Mel15].

Mostly even recently, all data segmentation is still interpreted as a two dimensional problem. The status of the literature up until very recently was solely to represent or transform 3D data into two dimensional representations in which the issue was solved, being the results extracted once again to the three dimensional spaces. On this regard, one of the biggest recent evolution steps taken on the field was the creation of PointNet. The work by Qi *et al.* [RQSMJG17] describes a neural network capable of completing scene parsing, object segmentation and classification using only raw, unordered point data. This methodology surpassed the previous state of the art accuracy for object detection in 3D data.

The basis idea behind this network is that the points in the cloud should be processed as unordered bundles on the network. This means that even if objects are rotated on the scene, they are still the same object to our network. In fact, it simply means that if the point cloud is considered as a set, then the network needs to be independent of any permutation of the set. This translates itself on a mathematical expression, presented in 2.3, that reflects the property of symmetry.

$$f(x_1,...,x_n) \equiv f(x_{\pi_1},...,x_{\pi_n}),\ x_i \in \mathbb{R}^D \tag{2.3}$$

The authors of PointNet realized then that the issue they had to tackle was to create a family of symmetric functions that could produce the same result to any permutation of the set of points as input. From these steps, they built a network which on a baseline transforms points into embeddings followed by a simple symmetric function which gives the output classification. The symmetric function that was reported to be the most effective was the maximum pooling function. Simply put, this network only embeds 3D unordered data into feature maps (data is segmented and partitioned) and then applies a maximum function on them for classification. Remarkably, this architecture was able to beat the state-of-the-art results for object detection using 3D data and is a huge remark that shows how the representation of data can be done differently through a more interesting and result achieving setting. Before PointNet the issue was interpreted as an identification of coordinates for bounding boxes. With PointNet, labeling is done for each point in the cloud which eventually reaches an entirely semantic segmentation of the space that had seldom been considered previously.

Overall, PointNet is a flexible network which can perform all three operations of scene semantic parsing (in which all objects in the scene are identified), object part identification (in which specific objects are divided in their own composing parts) and object classification (the objects are labeled from a wide range of classes). The fact that such a simple architecture can tackle so many problems with high effectiveness using only raw unordered, unprocessed 3D data proves that the investigation in the area is taking the right steps towards a system that can achieve this task reliably.

The pipeline of the classification network is as follows: first the points are transformed a couple of times by passing through one independent network which is called T-Net. This network has a similar architecture as the one previously explained, using max pooling and a multi-layer perceptron in order to apply transformations on the input cloud points. After the transformations are done, a perceptron extracts features from the processed point clouds. Max pooling is then applied in order to obtain the global features for the cloud. Finally, another perceptron outputs classification probabilities for the k possible classes using the global features.

In order to segment the scene, a few more steps are employed. Since segmenting the whole point cloud needs the reflection upon several different features that are both local (to each point) and global (to the whole scene), the segmentation network extracts the local point features through two perceptron layers with shared weights and joins them with the global ones obtained through the classification network. The output has then the size of nxm which represents both the points and objects dimensions.

On a recent paper, Charles Qi et al. have launched a variant network, the Frustum PointNet [QLW$^+$17]. On this work, a pipeline is proposed for the use of camera based detectors for search space reduction on RGB-D data.

A camera detector then feeds bounding boxes coordinates on 2D space alongside with a classification label. These flat coordinates are projected into the three dimensional space and a frustum is removed from the pointcloud. A dataset built from frustum areas is then fed to a segmentation PointNet followed by a residual center T-Net and a final amodal 3D box estimation network. This methodology still holds the podium on most categories of the KITTI benchmark for 2D and 3D object detection. One great achievement of this model is the several axis modifications suffered inside the frustum coordinates which make the data non sensible to rotations. Another one is the introduction of a corner loss which makes use of both the center and the bounding box coordinates together since they are naturally related on spatial environments.

Taking as ground work the region proposal network [GDDM14] and the PointNet methodology [RQSMJG17], some Apple researchers took it upon them to improve even further such techniques and present on their paper [ZT17] a network which they called VoxelNet. Much like PointNet, VoxelNet also works with raw, 3D data that serves as input to a feature learning network. This first network divides the space in voxels and then extracts the points belonging to those voxels into vectors with shape related information. Following this process, convolutional layers are added on top in order to aggregate the respective context of the working space. Finally, the region proposal network propose bounding boxes for all objects that have been found from the data.

Voxel representations have been widely used in deep learning approaches to 3D object detection. Voxels are grids that represent a three dimensional space and are specially oriented towards the representation of heterogeneous models. The grids of voxels may save a wide range of information such as color or height. These have been widely used in several applications and are specifically popular in the game graphics business.

Using convolutional networks, specifically, voxels are the most preferred representations to extract the 3D information for the network. Several works such as [MS15] and [JS16] have made use of the voxelized representation for object detection and classification with convolutional neural networks.

Also making use of the CNN, several laserscanner based methodologies have worked on widely ranging, area specific issues inside object detection and autonomous driving. For example, [MET17], a paper by Matti *et al.* adapted the region proposal network [GDDM14] to specific pedestrian detection with a 4-layer sparse laser sensor on the KITTI dataset[2]. Many other works have their attention on the use of convolutional neural network for urban areas specially for pedestrian detection. The work [SSJ06] is one of the oldest applications of these techniques.

Even more broadly on the topic of object detection for autonomous driving, convolutional networks have also been widely employed and experimented on. The work by Lange *et al.* [LUG16] describes the use of a standard convolutional neural network on 2D image based data. They compared several net topologies and concluded the best to be the following architecture: four convolutional layers with relu activation followed by max pooling (pooling layers include normalization afterwards), three fully connected layers with dropout in between and the use of gropping. The several topologies studied showed rather discerning outcomes which proved the impact that architectural design had on the use of these types of networks.

Li [Li16] also presents a possible approach for convolutional neural networks in which two output maps are described. On his fully convolutional network the activation chosen is the relu. The output maps, called objectness map and bounding box map, have the function of returning whether a region belongs to an object and the specific coordinates of the bounding box, respectively. The author points out the slower performance, consequence of executing 3D convolution when compared to the same operation in 2D. Following this issue, on another work, Li *et al.* propose the use of a 2D fully convolutional network to perform the tasks of bounding box and class label prediction simultaneously [LZX16] for 3D data. The network follows much the same structure, doing 3 different convolutions followed by deconvolutional layers which lead to the output of both maps.

Another CNN implementation for object detection in 3D LiDAR data, more specifically on the KITTI dataset, is the work by Asvadi *et al.* [AGP$^+$17]. The particularity of this paper is the reporting of the effects of data augmentation on the effectiveness of the network. The authors revealed an impactful improvement in performance when using such a technique to expand the training set.

2.1.4 Camera and LiDAR Fusion

Some authors like Oh *et al.* [OK17] have relied on using the fusion of the classification outputs obtained both on 3D LiDAR data and camera based approaches with convolutional layers. They also introduce region of interest pooling on each convolutional layer grouping together both color and semantic features.

[2]The KITTI Vision Benchmark Suite has datasets for 3D object detection available at `http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d`

Other kind of works have tried to tie the distance in between the 2D and 3D convolution. Liu *et al.* have proposed a volumetric spatial transformer network in order to do so in their work [LSZ+16]. The basic concept is to extract the depth information after doing a 3D convolution on a volumetric grid. The azimuth and polar angles are then used in order to extract the necessary depth information which leads to a use of the representation of the projection of the object on that surface. This is achieved with the help of a dense layer which computes the loss gradients of the depth representation (θ, ρ), azimuth and polar angles, with the standard back propagation method used in feed forward networks. Using the extracted features, finally, the 2D convolution is applied once again resulting in the outputting classification probabilities for all 40 considered classes. The activation used at the end is the soft-max loss.

Convolution seems to be the preferred methodology for object detection. However, some few works have reflected upon the use of recurrent neural networks for the task at hand. These types of works are specially directed towards an real time setting having the data being analyzed frame by frame as the sensor collects the data online.

On this purpose, the work of Prokhorov [Pro09] reflects exactly on the use of a recurrent neural network for real time laser data scanning. Having the data being inputted in a sequence allows the network to grasp the information in a temporal order. On this work, the problem is interpreted as having two classes only. This way, even if rarely, neural networks are employed for other tasks of object detection. These types of networks have been proved to be able to handle both sensory data in a real time approach. The studies of their applications have been moving towards that specific perspective.

2.2 Theoretical Foundations

Some models were toyed with specifically on the implementation so this section intends to enlighten the reader fully on the theoretical concepts behind those particular approaches.

2.2.1 You Only Look Once

The You Only Look Once model looks at the full image once. This was the original premise and with this, YOLO was able to efficiently extract context from the whole consumed image, joining both the global features and each object's specific characteristics. Taken from its original paper [RDGF16], image 2.2 shows the pipeline proposed initially for object detection. This model is a general purpose object recognition tool for color and grayscale images standardly.

As one can denote in Figure 2.2, the pipeline starts from the division of the image into an SxS grid. Each cell will then predict B bounding boxes along with a confidence score for each of them. A box confidence is evaluated over the multiple classes conditional probabilities and the overall intersection over union of the predicted box with the ground truth one. All classes are predicted at the same time for the whole image and that is the premise that makes the model as fast as it has always been. The resulting tensor encodes both the number of grids, the number of classes to classify and the number of bounding boxes that each cell predicts.

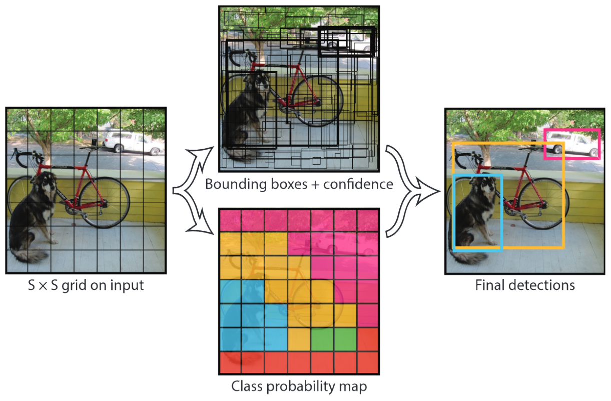


Figure 2.2: From the *You Only Look Once: Unified, Real-Time Object Detection*[RDGF16] paper, YOLO detection pipeline.

A cell on the grid is responsible for a detection and classification if the object's centre is found within it. A bounding box is then represented by 5 different variables: (x, y) is the tuple that represents the centre coordinates of the box, the (w, h) contain the width and height of the box according to the image proportion and a confidence value that the box contains the object. As such, the loss function is modelled over these 5 parameters, being a combination of the sum of squared errors with some characteristics added in order to balance errors in large or small boxes and distinguish between classification and localization mistakes. The full network architecture breakdown is described fully on annex A.1.

The whole YOLO model is implemented using the Darknet framework [Red16] and its classifier was previously trained on Imagenet [RDS$^+$15]. The architecture is quite simple, being a standard convolutional network with 24 layers followed by 2 fully connected layers. Non maximum suppression is also used in order to eliminate duplicated boxes. For a specific breakdown of darknet's architecture, refer to annex A.2.

Just like any other model, YOLO has its strengths and weaknesses. The incremental work done with the updates was in the sense of reducing the handicaps and augment even more the existent strengths. The most flagrant obstacle to tackle is the reduction of the sources of error from wrong localization. In a comparison in between Fast R-CNN and YOLO, Redmon et al. reported that whilst their network makes much less false positive classifications coming from background image areas, it looses most of its mAP on localization errors. Moreover, detection of small objects specially grouped together has been a great challenge to overcome.

2.2.2 YOLO Version 2

The 2016 paper [RF17] introduced the second incremental version of the model alongside a joint training algorithm that enables training on both detection and classification data namely YOLO9000.

On the newest architecture, some changes were made which focused directly on the region proposal setup, diminishing the number of location errors and the low recall achieved by its predecessor. All the while, the objective was to always maintain the classifier's accuracy. One direct point of approach was the difficulty that the original YOLO model had with small objects.

The incremental approaches taken are described below.

Batch normalization Its application on all convolutional layers improved mAP and stabilized the model. It also eliminated the need for dropout.

Higher resolution The classifier is tuned for 448x448 input resolution on ImageNet which helps adjust the filters for higher resolutions on the detection step. Opposed to the initial model which trained the classifier at a 224x224 resolution and used 448x448 for detection.

Anchors Inspired by Faster R-CNN's offset prediction, anchor boxes are used. Network resolution is reduced to 416x416 in order to have only one cell at the centre of the image. Higher resolution output after the removal of one pooling layer.

Clustering Anchor priors are chosen through $k-$means clustering. With a k=5, anchors are defined using a distance metric that involves the intersection of union between the box and its centroid.

Location predictions Bounding boxes are predicted according to the responsible cell. Sigmoid activation is used in order to constraint the ground truth to a range between 0 and 1. Parameters t_x, t_y, t_w, t_h, t_o are used to calculate the boxes centre/dimensions alongside with the top left corner offset coordinates (c_x, c_y) and the prior width and height p_w, p_h.

Fine-Grained Features Additional feature map of larger resolution (26x26) stacked with the 13x13 feature map through a pass-through layer.

Multi-Scale Training Training done with random net resolutions switched at each 10 batches.

Darknet-19 Classifier network changes to Darknet-19, a more mature and evolved model with 19 convolutional layers and 5 max pooling layers. Data augmentation, hue, saturation, crops, rotations and exposure shifts used in order to introduce variety in the data.

2.2.3 YOLO Version 3

Most recently, the YOLO project web page[3] has launched a newer update – version 3. On a preprint arXiv archive the changes are described through a technical report and the newer results are advertised as still being much faster than any model and on par with accuracy on the AP .50

[3]Found at https://pjreddie.com/darknet/yolo/.

metric. However, on the newer .95 metric, the model can not keep up with Retinanet in terms of accuracy, having around 7% less mAP. Throughout the analysis of the growth of YOLO, a fact immediately stands out. Even though this model seems to be great at localizing and classifying objects, it does not seem to be able to provide an exact, almost perfect bounding box that contains the object at its precise location.

The newer additions are generally explained on the following descriptors.

Objectness Objectness score was introduced in version 2. However, on the newest update, this score is calculated using logistic regression which represents the ratio of overlapping in between the ground truth object and the bounding boxes prior.

Classification Softmax is removed and independent logistic classifiers are added instead. Predictions are then obtained using binary cross-entropy.

Pyramid extraction Much like feature pyramid networks [LGG+17], YOLOv3 extracts features at several scales and concatenates the upsampled feature maps with ones obtained on earlier stages.

Anchors Box priors are still extracted through $k-$means clustering, only now with 9 clusters and 3 arbitrary scales.

Darknet-53 The classifying network evolves towards 53 layers and adds shortcut connections in order to process the connected feature maps with more fine grained information. This network is also trained on ImageNet [RDS+15] and achieves top classification on par with ResNet-152 but much faster. The Darknet-53 architecture is described on annex A.2.

With these newer updates, the model moves towards more accuracy. Even whilst loosing some speed, it is still a very powerful and fast network. It is reported that small objects are not such a challenge for the model anymore however, it seems that more difficulties arise on the larger and medium objects.

2.2.4 Focal Loss

Being used specifically on chapters to follow, the focal loss introduced by FAIR [LGG+17] is of interest for the total understanding of the concepts underlying some implementation choices. Having been developed with the intent to balance foreground and background classes on one stage image based object detectors, the focal loss primes in applications for classification with unbalanced data.

As it has been described on the paper *Focal Loss for Dense Object Detection*, the focal loss is derived from the cross entropy loss for binary classification. The CE loss is showcased through equation 2.4 and described as the negative log of the probability of the active class.

$$CE(p,y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{if } y = 0 \end{cases} \tag{2.4}$$

On equation 2.4, the p corresponds to the probability of the class being classified as active when the ground truth is active as well. From this loss function, one fundamental issue can be denoted. When there are many easy examples, meaning a sample can be easily classified, its loss value will be small. However, if many samples are easily classified, these even though being small values, when summed will still overwhelm the class that would be considered as rare. For this reason, two balancing parameters are then introduced. One is mentioned on the paper as a modulating factor, an exponent applied on the opposing probability. The other is more commonly used on many other cases, a regularization factor multiplied by the loss value.

Compiling these concepts gave rise to the focal loss formula, presented through equation 2.5 where p_t represents both the probability of active or non active classification in respect to their labels. Through the use of this loss, the dominant class does not overwhelm the gradient as the value it would adopt on CE is downsized through the γ factor operating on the opposite class. The α provides more balance to the loss and improves stability.

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t) \tag{2.5}$$

2.3 Overview of Datasets and Benchmarks for Object Detection

One of the most widely known and research used benchmark for object detection in 3D data is the KITTI Vision Benchmark Suite [GLU12]. Built using a test vehicle that has collected data in real world varied urban locations, this tool provides raw datasets accompanied by benchmarks and the availability of performance evaluation on their website platform[4]. The datasets and benchmarking options range from the areas of stereo, optical flow, visual odometry, object detection both in 2D and 3D and tracking of objects. The evaluation for 3D object detection scores are carried on through the evaluation of results with three different standards: easy, medium and hard. These levels are determined according to the bounding boxes height, occlusion level and truncation. Overall, this benchmark expects 70% of overlap on the bounding boxes for cars and an overlap of 50% for cyclists and pedestrians. The collection of 3D data is carried on with a Velodyne HDL 64E.

Another recent benchmarking tool which revolves around an interesting dataset for both 2D and 3D LiDAR point and RGB camera classification is the Semantic3D.Net [HSL+17] dataset. Contrary to the representation required in KITTI which uses bounding boxes, Semantic3D.Net

[4]KITTI Vision Benchmark Suite's website allows the upload of the results obtained on their datasets at `http://www.cvlibs.net/datasets/kitti/index.php`.

interprets each point separately, labeling each and every single one with a color that represents its class. This approach is the closer to the mindset used in PointNet which estimates labels by points and not by bounding box (the so called semantic segmentation). With the results shown by this network, it seems plausible to think that this approach to representation and labeling might be just as effective or perhaps in the future even more interesting for object detection[5].

On the topic of broad datasets, there are several ones that have been quite important to the evolution on the field of 2D and 3D based detection alike. Even if most of these have a general purpose kind of scope, many do contain certain classes such as vehicles. Furthermore, it is important to denote that there is also work done on other types of 3D based data. Some efforts on this direction are then contemplatively named.

Working with three dimensional digital based models, the Princeton Shape Benchmark [04] was one of the first to surface on the field. Offering a wide range of objects to classify, the objective was to allow evaluation of shape-based retrieval and analysis algorithms.

Recently, Princeton university has also created a benchmarking dataset called ModelNet[6] [WSK+15]. This benchmark is based off of 3D models extracted from CAD. PointNet was one of the networks to be developed under this benchmark and is currently ranked on the ModelNet10 and ModelNet40 datasets for object classification. The dataset has 662 object categories available. The models presented however are not indicated for autonomous driving since the majority of the models represent objects found in indoor locations.

Under the PASCAL name, there are interesting datasets and benchmarks to be found. The PASCAL VOC [EEV+14] is a long lasting project which aimed at providing standardized image data sets for object class recognition. More contemporaneously, the PASCAL3D+ [XMS14] launched. Being characterized as a three dimensional based benchmark for object detection and pose estimation, it features a panoply of CAD models including classes such as vehicles, bikes and bicycles.

The recent *Large Dataset of Object Scans* [CZMK16] featured an interesting collection process. Having publicly released both RGBD scans and 3D based models, Choi et al. asked common workers to scan objects of their daily lives. This resulted on a huge, greatly varied bundle of data made available for object detection and classification research.

2.4 Summary

Object recognition techniques can be divided in two major groups: geometrical or deep learning driven. The geometrically oriented techniques focus on the extraction of features by hand related to the objects geometric properties such as shape detection, fitting of line segments, grouping of points and many more. On LiDAR, these techniques can be grouped under four major basis: parametric, clustering, occupancy grids and breakpoints. In methodologies grouped under parametric the approach is to fit the points to known basic shapes. On this group we find algorithms such as

[5]Dataset and documentation can be found at http://www.semantic3d.net/.
[6]ModelNet available at http://modelnet.cs.princeton.edu/.

RANSAC [FB81], Split and merge and iterative end point fit for example. On the other hand, clustering is based off of the idea that points can be grouped and these groupings represent objects. The radially bounded clustering ([KWB08] and [PJ17]) is one of the techniques that fall under this category. Occupancy grids have a huge importance in the field as much as an object detection tool, used as much for points segmentation and for preprocessing for the use of deep learning. Their representational power gave them an advantageous position in the field, being used in order to depict the object's three dimensional properties. Finally, breakpoints explore the intervals in between segments. On camera detection, simple visual features were extracted such as color and texture. These methodologies were archaic and non adaptable to new scenarios. The field only suffered evolution once the power of CNN was discovered.

On the field of deep learning, the use of 3D data is still very limited. Most work focuses on the extraction of non oriented bounding boxes from camera images. On the state-of-the-art, from what was found, only one methodology - PointNet [RQSMJG17] - works with real, raw 3D data. Other approaches convert or transform the three dimensionality into a two dimensional issue which then is extracted back to the original dimension. The most widely popularized used deep learning techniques for object detection are convolutional neural networks. Several architectures and settings have been explored in order to achieve higher accuracy levels, portability and speed. Recurrent neural networks have also been applied on a very few cases more specifically on real time settings. Both 2D and 3D based approaches count on the use of the convolution as their main process of feature extraction.

The labeling representation is also a step that has proved to be important for the end results of deep learning in object detection. Most conventional approaches expect the identification of bounding boxes and classify objects within those boxes. However, recent works have explored a point by point labeling which enables the techniques to extract both local and global features. Approaches on this area have proved to be as good or even better sometimes at detecting and labeling objects. Even though the research seems to be on the right track, there is still much space for exploration for object detection in 3D environments. For camera detection, the field seems to have matured almost as much as desirable. Even so, improvements can still be made.

On the topic of camera detection, a bundle of algorithms are explained. One of the most important ones, the R-CNN [RDS+15] is extensively demonstrated even through the incremental process it suffered. The two categories of object detectors are also delineated, featuring the most impactful models that helped bring the field forward. That is the case of the SSD [LSZ+16], the YOLO [RFA18] and the RetinaNet [LGG+17]. The operational methods of these models are also described.

The theoretical concepts that were found to be the most important for the implemented solution are laid out as well. Starting from the description of the YOLO model in depth, the incremental process in between its versions over the years is delineated. Besides this, the focal loss is explained as a derivation over the binary cross entropy loss that facilitates training of deep learning models in cases that feature heavy class imbalance.

An overall summary of datasets for research and evaluation of methodologies for object detection in both 2D and 3D is also carried out. Works are mentioned as much on the general plane as on the specific, autonomous driving efforts orientated.

Chapter 3

Machine Vision and Autonomous Driving

In order to give a vehicle the ability to perceive its surrounding environment, several differently wired sensors are made use of for special, specific tasks. The two most popular sensors to be found on today's autonomous driving vehicles are the RGB cameras and the LiDAR. These two can give an accurate perception of the scene the vehicle is inserted in, leveraging from two dimensional representations to three dimensional, more geometrical intuitive settings. As such, these are widely popular and invaluable for most tasks required by autonomous driving scenarios. The work at hand focuses on the tasks of object detection using the designated sensors separatly or jointly. Even if other types of hardware, such as radars, GPS, proximity sensors and others are also necessary and widely applied to contemporary vehicles, the two chosen sensors provide a great tool for they do the biggest bulk of the task of construction of both 2D and 3D environments. These sensors, alone, can facilitate the task of object detection and it is desired that the future of investigation on the area leads to the shrinking number of employed sensors on such vehicles, preferably using only cameras and/or LiDAR.

Following up, this section describes methods of operation for both sensors and develops on the concept that this work aims at studying in order to help further the efforts towards autonomous driving. The description of the sensors processing methodology is not deepened as the study of sensors is not a goal of this project. Instead, these general considerations are presented on this section only to enlighten on the capability of information retrieval of each sensor, with the objective in mind that the shortcomings and advantages of each need to be leveraged through software as best as possible in order to arrive at the best, most accurate result pool.

On the case of autonomous driving, the topic is broad and the issues to be solved are overwhelmingly challenging, all the while with these requiring a certain, quantifiable guarantee of the safety level that is required for such critical systems. An introduction to the specific object of study is then ensued.

Table 3.1: Sensory setup of the test vehicle from KITTI's dataset project.

Sensor	Type	Quantity	Available data
Velodyne HDL-64E	**Laserscanner**	**1**	**Pointcloud**
Point Grey Flea 2	**RGB Camera**	**2**	**Left/right frames**
Point Grey Flea 2	Grayscale Camera	2	Left/right frames
Edmund Optics NT59-917	Varifocal lenses	4	-
OXTS RT 3003	Navigation	1	Coordinates

3.1 The Concept of Sensory Perception

Machine vision has become more and more popular on the contemporaneous field research. In fact, terms such as sensor fusion and sensory perception are all the rage as much in companies as in final consumers. The efforts towards the construction of vehicles that can smartly and accurately understand their environment is a dream of many generations that seems to be closer each second as research stumbles and pushes forward the concept. The sensory perception idea entails the mix of hardware and software that can give a vehicle an accurate and valuable representation of the world. It entails invaluable knowledge such as the interpretation of the vehicle's own global and local position relatively to what surrounds it, may it be other vehicles, pedestrians or natural and artificial obstacles.

At the center of it all, the sensors are the foundation of perception. How they collect, what they collect, in which conditions and the percentage of workable information that they can produce are the basic questions that forge the line between a well built perceptive vehicle and an uninspired failure. As such, it becomes urgent to evaluate the hardware used for sensory perception through its fundamental description and the comparative analysis of their standalone strengths and shortcomings as well as their possible outcomes when fused together.

3.1.1 KITTI Vision Dataset Setup

Since the work at hand did not intend to build a reliable data source collected from these sensors, in order to shift the focus directly towards the issue, a renowned dataset built for autonomous driving research was used. Having both camera and LiDAR information available for investigation characterizing the same frames, this collection of data was incredible useful as an input measure. The test vehicle used to collect the information had specific versions of the discussed sensors and as such, the particular setup described on the KITTI paper [GLU12] is discussed on table 3.1. As can be seen, the setup vehicle contains a location system, the GPS and varifocal lenses for focusing purposes. Besides that, the standard RGB and grayscale cameras are also present. Finally, the LiDAR sensor used is one of the most powerful by company Velodyne.

Even if the principal sensors in study the RGB camera and LiDAR, some experiments described on further sections were run on the grayscale camera images (entry 3 on the table). These

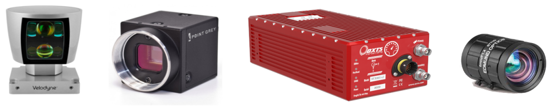


Figure 3.1: Sensors used by KITTI's team to collect the data.

particular experiments are rightfully identified as produced on the grayscale frames. Future research on this topic could include the exploration of sensory fusion between RGB and Grey images in order to develop a deep learning training platform for object detection in night or extreme conditions.

3.1.2 RGB Cameras

In summary, RGB cameras are produced and designed following the human eye mode of operation. Even if not totally in pair with the quality of the human eye, most cameras nowadays can capture the photographed scenes almost as well as a human looking directly at it would. This means that, in the spectrum of visible light, colors and white balances are collected, having environments described through RGB pixels.

3.1.2.1 Method of Operation

RGB cameras have diverged into groups accordingly to their method of operation. However, most RGB cameras fundamentally operate using four main, invaluable components. Those can be listed as a lens, an infrared filter, color filters and a sensor.

The pipeline of the capture of a color image is illustrated through the component directional representation found on figure 3.2. When the light is captured by the lens, it goes through an infrared filter and following that, in RGB cameras, there is also a color filter in order to extract color information from the scene. The sensor brings everything together by creating and balancing the information received, outputting a signal for transformation into a digital, standard image.

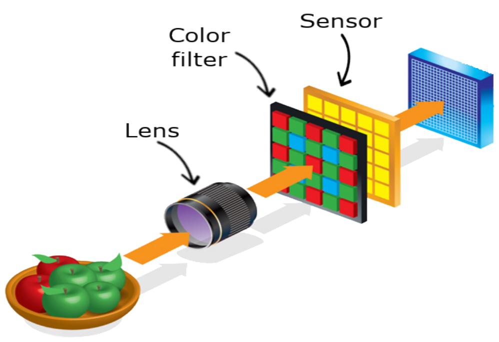


Figure 3.2: RGB camera image capture pipeline. Adapted from Digital Camera World's article *Cheat Sheet: how your digital camera turns captured light into an image* [Wor12].

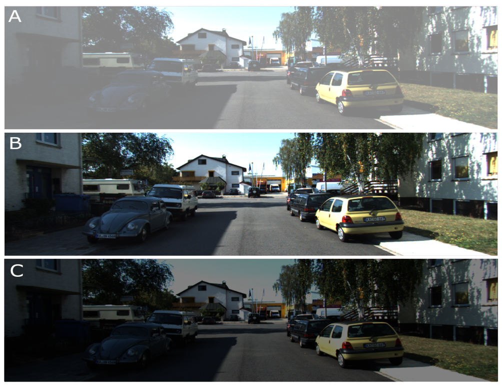


Figure 3.3: Examples of different white balances.

A camera can capture anything so long as it is on the scenario facing the lens covered by the zooming space supported. The lens incorporated will always influence greatly the outcome, since this may mean less or more horizontal field of view, directed or univalent focus and many other characteristics. The importance of the sensor can not be disregarded either since it is responsible for understanding the light conditions and work through them in order to balance the whites of the final picture. Figure 3.3 showcases two examples of incorrect white balancing on A and C. Only picture B has the correct light setting.

Even if these components create the main working pipeline, more specific parts are needed in order to ensure the retrieval of information that would be successfully processed afterwards. This is even more noticeable on the mounting of these types of sensors on vehicles. Such mechanisms are needed for example to help stabilize the image and to ensure that the picture is focused over different depth ranges. This usually means the completion of such setups with other kinds of components such as the varifocal lenses used on the KITTI's test vehicle described on table 3.1.

The importance of the effectiveness of these components is demonstrated through some examples of the output that badly mounted systems can bring out. Figure 3.4 showcases two samples from the object dataset with two different focusing techniques. A is focused on the global scene whilst B is focused on the traffic sign noticeable on the right which makes the identification of the vehicle more challenging. On the other hand, figure 3.5 exemplifies the noise introduced through an uneducated ISO regulation. These examples are somewhat in over proportion however, the dan-

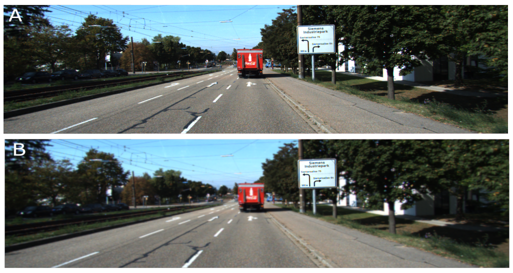


Figure 3.4: Effects of focus changes.

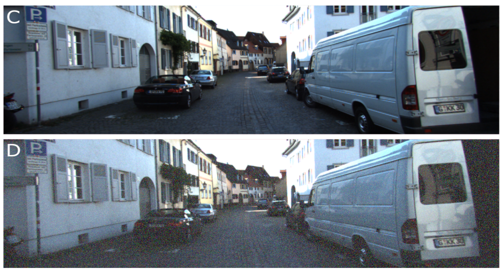


Figure 3.5: Effects of ISO regulation.

gers of wrongly setup RGB cameras are still existent and may influence the software interpretation on the final end.

3.1.2.2 RGB Cameras and Autonomous Driving Scenarios

RGB cameras compile a flat representation of the three dimensional world which is close to the apparently flat observations extracted by the human eye on a similar way. The fact is that research work on such data is made easier as it is faster for humans to abstract geometrical properties based on the methodology they make use of in daily life. This is proved by the huge amount of work found on the area of 2D object detection and the advancements made through those efforts. Besides this, processing computationally images is rather fast and efficient nowadays which makes RGB bi-dimensional data a great candidate for speedy achievements in 2D.

The most notable advantage brought by RGB cameras over grayscale is the fact that each pixel contains 3 channels of information - color. This is useful for more meaningful extraction of data and the use of color might be a factor that improves performance on neural networks.

Moreover, humans have been driving vehicles around for many decades and their own sensory perception has allowed a somewhat safe navigation in such environments. However, humans are armed with more perceptional tools and intelligence to process such information than current camera fed object detection algorithms are. Even if a human does not see a certain object, it can still identify it through sound, shadows, touch or even smell. These special abilities are not shared by machines yet which makes this a sort of shortcoming on only using cameras for road based object detection.

A more pressing issue is that such sensors suffer greatly with inadequate weather conditions. Even the best existing sensors can not cope well enough with these scenarios delivering images that may be deemed useless for object detection and algorithmic processing. Fundamentally, an RGB camera needs light in order to operate. This limits its applications on the road from the starting point since driving occurs at any time of day or night and at any environmental condition.

Besides this, cameras have the handicap of capturing only what is in zooming range of the lens. This means that, if the objective is to give a vehicle a 360 degree view of its environment using only cameras, it is necessary to have a panoply of these sensors in different positions which leads to a more complex setup.

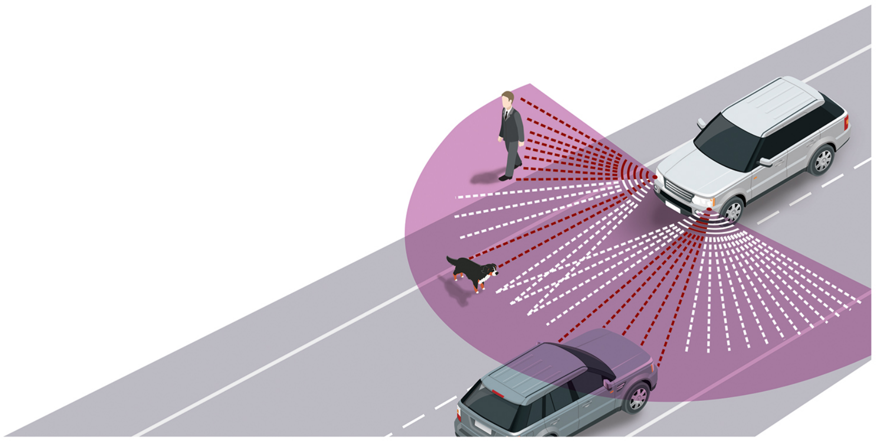


Figure 3.6: LiDAR data collection process example.
[*] Image by Guest Post on ValueWalk's "Why Tesla Motors Inc.' Autopilot is Fundamentally Flawed".

Further perception of the scenario is not possible in only 2D settings. Whether the work is done in segmentation masks or bounding boxes, the outcome is bi-dimensional like its data so no information about the object's depth is available through these methodologies. This eventually means that a vehicle can not navigate safely as it does not understand with confidence whether an object is up close and in dangerous range or if it is far away with no impact on the vehicle's immediate field of action. It also limits the guidance in situations such as overtaking long vehicles.

In order to avoid such lack of important information, some investigators wonder about the possibility of use of depth cameras which also give a depth distancing factor. However, more research on the applicability of this process is necessary to provide a defining conclusion. Besides this, depth based cameras are outside of the present scope thus are not considered.

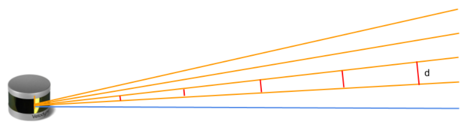
3.1.3 LiDAR Sensor

Opposed to camera sensors, the objective behind the development and inclusion of LiDAR sensors on current autonomous driving vehicles is the collection of three dimensional data. Based on the principle that, upon being hit with an electromagnetic wave, any object will produce a reflection, these sensors are able to determine the distance at which surrounding objects encounter themselves from the sensor, building a point cloud with a three coordinate axis representation.

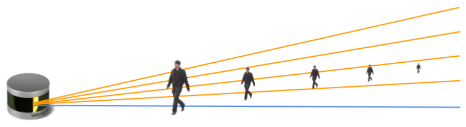
3.1.3.1 Method of Operation

LiDAR fundamentally works through the emission of laser pulses at a predetermined frequency. When these pulses hit the objects around it, the material reflects the pulse back to the sensor. Using the readings for the travelling time between the discharge of the laser pulse and the reception of its reflection on the sensor, a point cloud can be built in three dimensions which accurately depicts the environment.

Figure 3.6 shows an example of the functioning of the sensor. When the laser beams do not hit an object, the point cloud will produce empty space on the respective area of the pointcloud.



(a) Distance between beams increases with range.



(b) A person can "disappear" from the data as distance to the sensor increases.

Figure 3.7: Some limitations of the LiDAR sensors.

When the beam finds itself on an object (red lines), the reflection is collected and the respective points are added to the point cloud.

Just like any other sensor, LiDAR has its own shortcomings and obstacles that are still a research topic to this day. The sensor in question can build more or less dense point clouds depending on the specific limitations coming from its hardware. The sensor has a **shadow zone** in which objects are not easily detected. This happens due to the fact that as the horizontal distance gets larger, the angle in between laser beams leads to a larger distance in between them. For example, looking at figure 3.7a, if we consider that this sensor[1] only outputs 8 laser beams (only the upward side is shown), it can easily be seen that the distance (represented in red) in between the beams becomes larger as the distance to the sensor enlarges. In fact, taking a look at 3.7b, one can conclude that a person may "disappear" from the point cloud since as the distance to the sensor is larger, a shadow area surfaces that is larger than the size of the person on that perspective.

Furthermore, another known issue related to LiDAR object detection surfaces when black objects are considered. Since these types of items absorb most of the laser beams received, the point cloud becomes sparse and sometimes it becomes illegible to recognize the existence of objects even with the human perception. This may lead to various types of objects (even vehicles) to be occluded and missed from the object detection process.

3.1.3.2 LiDAR and Autonomous Driving Scenarios

The premise of the laser scanner type of sensor is to provide information that no other sensor is able to in autonomous driving scenarios. Even if it is not fully reliable at all times due to the properties showcased on the previous section, it has the power to build a uniform, three dimensional, 360°representation of the surroundings. The intricacies found in a three axis based interpretation still have a lot more exploration to be done. However, it is the researchers belief that this could leverage real strength in road scenarios so long as software becomes more evolved and able to process 3D based data in more efficient ways.

[1]Version represented on the pictures is the Velodyne VLP-16 PUCK.

Table 3.2: Constraints of the RGB Camera and the LiDAR.

Sensor	**Constraints**				
	Weather	**Object**	**Range**	**Time**	**Field of view**
RGB Camera	Light	-	Resolution	-	Zoom space
LiDAR	-	Black materials	Point density	Refresh times	-

Another prospect of using LiDAR on these environments is the fact that it can present a view of the entire surroundings with the single use of a sensor. Even if it is less complex in terms of setup (since a panoply of cameras needs synchronization and calibration procedures), LiDAR sensors are known to be rather expensive and the better they are able to represent the environment reliably, the more pricey they get.

Furthermore, the limitations intrinsic to the hardware explained on the previous sections may lead to failures in object detection procedures that are not due to the own software's accuracy. These types of situations make it harder to ascertain the model's efficacy. This is another reason to use a processed dataset that already has filtered out objects with heavy occlusion from the precision measurement. It makes the process of enhancing models all the more strong due to the correct and reliable tracing of errors.

3.1.4 Comparative Analysis of RGB Camera and LiDAR

As it has been previously discussed, both sensors in discussion leverage advantages as well as some shortcomings. Overall, it is of interest to consider a comparison of the extracted data and their possible explorations when in application on the issue at hand. Table 3.2 describes the constraints found on each sensor. When employing such hardware in autonomous driving scenarios, these constraints need to be taken in consideration in order to ensure the reliability of the end results. As can be denoted on table 3.2, both sensors suffer as distance from the vehicle increases. Pointclouds become sparser the further the object is and in images, their representation becomes a small bundle of pixels with weak resolution. On the other hand, cameras suffer from weather conditions that entail lack of light or diminished lightning. LiDAR has the drawback of badly representing black objects. Furthermore, LiDAR has a refresh interval in between each scan which means that it can not provide a 360°view as fast as a setup of several cameras would. Approximately, cameras can be three times faster than a LiDAR sensor. However, in order to achieve the whole environment view, the several cameras need to be configured so that their scope covers the whole setting.

On the topic of information leveraged by both sensors, table 3.3 showcases the different possible features to be found on data provided by each sensor. Both sensors extract width and height information even if on different planes. LiDAR has the advantage of providing a depth coordinate which gives a better understanding of the object's location on projected 3D space.

RGB cameras provide data related to textures and color of objects which is quite useful information for convolutional neural networks to extract. The fact that, on the 2D plane, objects

Table 3.3: Information that can be extracted by both sensors and some applications.

Sensor	Contained features	Principal additional applications
RGB Camera	Texture Color Height Width Shape	Traffic lights and signs recognition Lane detection Object tracking Semantic and instance segmentation
LiDAR	Intensity Material Spatial distribution Proximity Height Width Depth	Orientation estimation Object tracking 3D Semantic segmentation

mostly keep their shape intact (with some dimensional variations) is also an advantageous factor that networks may learn to identify.

On the other hand, LiDAR contains intensity and material representations. Due to the collection of reflected rays, physical properties are intuitively derived. This sensor is also useful in order to extract proximity relations in between the vehicle and the space around it. Due to the three dimensional distribution of LiDAR data, the spacial relations inside objects and between them are more easily described. These can be a decisive factor for networks of the future.

3.2 The Problem of Object Detection in Autonomous Driving

The issue of object detection is quite wide in application and range. For such reason, it would be of interest to consider this issue on a research based perspective when applied to autonomous driving settings. Furthermore, there are several constraints that limit the scope of this work at hand. It is also important to lay them out, alongside with the obstacles to be faced when applying these methodologies on the real world scenarios.

3.2.1 Autonomous Driving

The core issue addressed by this work is, on the first plane, that of autonomous driving. Assisted driving systems reflect certain levels of automation of the vehicles. Lower levels indicate that the system aids the driver in some driving tasks. However, the human driver is still very much essential to the process. On the last level, an autonomous driving system is defined as an unmanned process in which a vehicle can safely understand its surrounding environment and make intelligent decisions in order to guide its passengers from their starting point to their destination.

With autonomous driving gaining its momentum in popularity, several semi autonomous or even fully autonomous driving vehicles have started to be tested on all kinds of roads. Even with all the optimistic results achieved by these vehicles, their 100% safety is still not assured. Recent developments in the industry have seen these kinds of vehicles be involved in polemic due to their inability to prevent some accidents. This means that there is still more on the road to be paved in order to achieve the dream of a widely popularized use of these types of vehicles or transportation systems.

On current setups for autonomous driving experiences, a panoply of sensors that measure important information to assist driving processes are present. The most relevant categories of sensors are defined below:

LiDAR sensor 3D information gathering sensor.

GPS Synchronize the vehicle's position on the internal location map.

Camera 2D information gathering sensor.

Distance sensor Measure distance to immediate objects.

The leverage of such complex mixture of sensors implies processes of sensor fusion and other relevant techniques that are not fully covered on this work. However, it is to be denoted the high price that such systems still entail and the lack of completely reliable processes that ensure the elimination of precision uncertainty. This work discusses sensor fusion or similar processes on the software level and aims at concluding how these sensors can be of help to each other as much as a certainty reduction methodology but as a platform to diminish search spaces.

3.2.1.1 Object Detection

Autonomous driving includes a wide range of complicated issues and processes that provide challenges of great research interest. The focus of this work is, however, solely on the phase of object detection. Object detection is the process in which the vehicle collects all kinds of data on the environment and uses this data to recognize the existence of all objects surrounding it. The final goal of object detection is to ensure that the vehicle makes the best driving decisions at all times and, even on the topic, more specialized topics can be approached such as distinguishing driving from non driving areas on the path planning module.

Object detection goes hand in hand with the topics of object identification and recognition. Diagram 3.8 showcases the relation in between the object areas mentioned. As it can be denoted, object recognition involves both processes of finding the object's location and classifying it into a group. On the other hand, object detection and object identification are specific tasks inside the area of object recognition in which objects location's are found and are classified, respectively. This work is presented as an object detection project since it does not focus specifically on classifying objects correctly. In fact, the most explored methodologies intend to detect objects through the use of a bounding box limiting their predicted location. Even so, classification is also part of the ensemble of such algorithms that are to be described on following sections. However, the sole focus was to have a system capable of understanding that *there is* an object *there.*

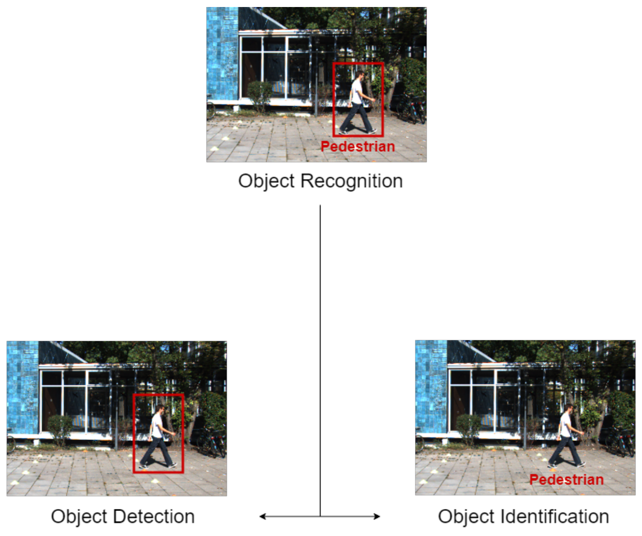


Figure 3.8: Relationship between object recognition, detection and identification.

Once object detection becomes robust enough, classifier networks can then confidently report that the object there is *this*. Since this project aimed at understanding the foundations of object recognition, object detection becomes the evident starting direction of exploration.

3.2.2 Constraints to the Problem Formulation

The issue at hand is very broad in terms of its practical application. So as to reduce the complexity to the available time constraints, the problem will be approached in an exploratory sense. This work intends to test different deep learning techniques for object detection in several environments with different constraints. Even if machine learning can be a power house in such issues as well, this work tens to focus more on variations of deep learning techniques. Through this project, experiments are digested and performed with a curious mindset that wants to understand what would be an interesting future research pathway or not.

Due to the decreased study of classification of objects, only critical classes of objects are considered. Critical classes can be explained as the groups of objects that are most frequent on roads and the ones for which safety is more pressing to be ensured at all times. The classes of interest in most experiments are then **Car**, **Pedestrian** and **Cyclist**. On some experiments, more classes are included, however, for metric evaluation purposes, these are the three main groups to be considered.

Furthermore, due to the difficulty of sourcing both camera and LiDAR information, only the dataset available for research from KITTI is used at all times even if on different ways. Real world testing is not considered even if it is a future objective. Besides this, some constraints come as standing principles from the dataset's own benchmark methodology. From the start, evaluated bounding boxes have a minimum size of 25 pixels. This ultimately means that any accuracy metrics explored in any experiment do not take into account predicted boxes below this size pool. No false positives are considered either from this behaviour. An intersection of union overlap is not considered whilst evaluating the performance of the implemented models. However, on the benchmark, only boxes with minimum overlap of 70% and 50% for cars and pedestrians/cyclists respectively are counted. Don't care areas with labels are not imported even for training operations. These are also not included in evaluation for the benchmark. Finally, the division in difficulties presented is not considered on experiments. On the contrary, all data is used for training and testing purposes equally. This is taken into account when remarks are made on the modeled solution's accuracy. Accuracy over difficulty levels is exposed in some sections that contain benchmark information.

On real world scenarios these constraints are not found. However, in order to have a better understanding of what an accuracy value truly signifies, these aspects are taken into account. This helps the process of result comparison and effective critical analysis. When the problem is extracted to more complex, world-like settings, the metrics must be carefully adapted in order to extract meaningful knowledge.

The use of cameras alongside LiDAR creates many calibration issues. The most notorious one would be the fact that the dataset only presents the front view of the vehicle on camera images whilst LiDAR has a 360 °view of the environment. For this reason, only the front view is considered whether in mixed sensory information experiments or in standalone LiDAR settings.

3.2.3 Obstacles to Object Detection in Autonomous Driving Scenarios

Even with the leverage of powerful sensors and techniques, challenging situations will always arise on the road. A perfect system would be able to predict every single outcome at every fraction of second as the scenarios unfolded on the road. However, even with the greatest hardware aligned with the best software, the roadblocks to overcome are many still to achieve this dream-like talked-of ensemble.

One of the major issues that object detection in autonomous driving faces is the fact that all methodologies applied need to be computationally solved at an incredible speed rate all the while leaving a small memory footprint. Employing such technology on a vehicle means that the processing power is limited to the existent, closed system and the software has to deal with faults in a way that does not put internal and external users at risk. Furthermore, inference must be as fast and as accurate as possible. Even if the vehicle is capable of understanding its environment with perfect precision, if the working time of each cycle takes more than one fraction of a second, the system will fail to ensure safety. The same issue will be faced when the roles are reversed.

Moreover, even if a model is trained to recognize an astounding group of limitless classes, new objects that the model does not have a reference point on will eventually start appearing on the road. This would happen for example with the introduction of new cars with fundamental differences in design (as it is expected to happen in the future). This raises a question that would risk the durability of such systems. How to viably update these models in a way that they keep old knowledge mixed with the new introduced, learnt shapes? And, more importantly, how to quantify the quality that each model has in metric accuracy after it has suffered such increments? These are still on going research topics with many more questions than answers still.

On the topic of datasets, there is also the introduced uncertainty of whether the used training sets are representative enough of the real world. Evaluating these kinds of systems on laboratory, human made sets can give a reliable confidence that the model will transport well to the application settings? If not, can these systems be safely tested on real roads?

As these vehicles start populating the road, sometimes with disastrous results, more obstacles start rising for autonomous driving. Failed experiments in public set back the field and there are still many unanswered moral, social and technological questions on the topic. All these are obstacles to object detection itself for, from a certain point, there is no possibility of reaching new conclusions if no new, out-of-the-box steps are taken.

3.2.4 Metric Discussion

A bundle of metrics were made use of in order to evaluate the quality of the results. Even if calculated through the same methodologies, these metrics extract different meanings for each task at hand. The theoretical foundations of each metric are laid out below.

All metrics rely on the interpretation of outputted labels as true or false positives and of true or false negatives. These categories are relative to a class and can build a confusion matrix. The confusion matrix is a notion that simply compiles these categories on the form of a matrix structure. These concepts are laid out on figure 3.9 for an example of binary classification.

True positive of class 1 The sample is classified as belonging to class 1 correctly.

True negative of class 1 The sample is classified as class 0 correctly.

False positive of class 1 The sample is classified as class 1 when it actually is 0.

False negative of class 1 The sample is classified as class 0 when it belongs to class 1.

As one can denote on figure 3.9, there are borders in between the relevant elements and the selected elements. In relation to a class, on this case would be class 1, there are a bundle of elements correctly belonging to it. If our model can classify all of them correctly, the border of the selected elements will be situated at the relevant elements corner. On the figure, the samples are evenly split through the various categories. However, it rarely ever happens on real training situations. The borders move as the classifier becomes better or worse at labeling samples.

The use of false positives and true negatives is necessary in order to understand whether our model classifies not only the existence of the class of interest but also its nonexistence. If our model would classify all samples as 1, then it would have as many true positives as relevant elements which eventually means it has learned to classify well all the elements in study. However, if we take a look at the other side of the categories, it would become easy to understand the model has learnt nothing at all, thinking all samples belong to the class at hand.

Through the use of these concepts, correct assumptions can be truly made on the quality of the model's results.

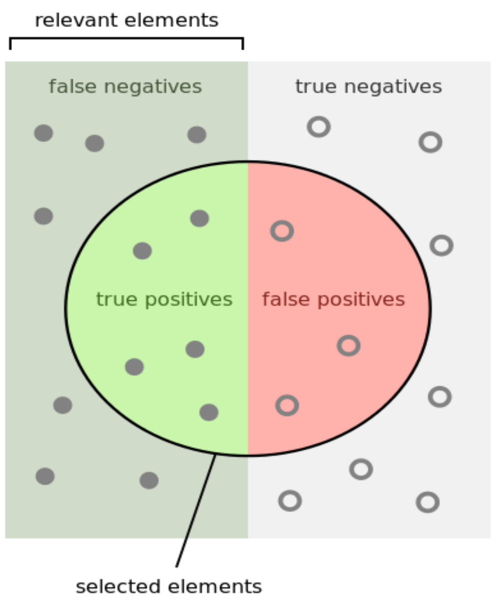


Figure 3.9: Binary classification metric analysis breakdown. Adapted from [Wik18].

3.2.5 Recall, Precision and F1

The recall metric is directly related with the precision and f1 score. These metrics operate on the distribution of the confusion matrix. Figure 3.10 showcases the areas that each represent out of the example representation on 3.9.

On binary classification, **precision** then represents the ratio of true positives out of all the selected elements. This value varies in between zero and one. If it is zero, it means that all the selected elements are false positives and if it assumes one, all selected elements are true positives.

On the other hand, **recall** signifies the amount of true positives in relation to the amount of false negatives. When this value takes its maximum (1), it translates the fact that the model selects all relevant elements. If it reaches zero, then no relevant elements are selected by the model.

On the case of segmentation of interest areas, recall being over a certain threshold is more important than precision being high. This behaviour would signify that the model can select a high rate of relevant elements even if it classified some non active pixels as active.

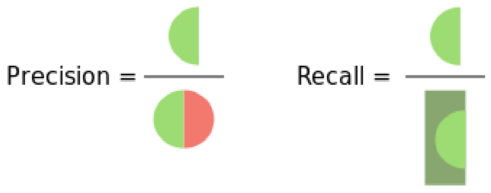


Figure 3.10: Recall and precision as seen from 3.9 in a binary classification scenario. Adapted from [Wik18].

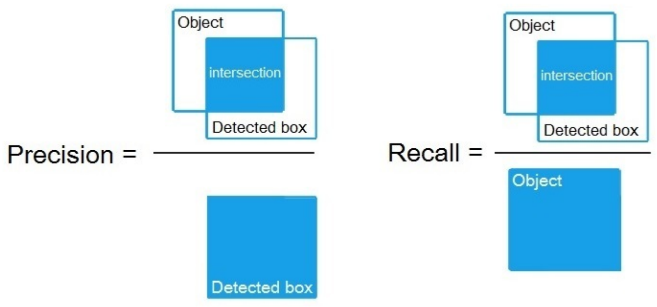


Figure 3.11: Recall and precision in a bounding box regression scenario. Adapted from [Ale18].

On the case of bounding box prediction, however, the metrics of recall and precision take on a new representation. Figure 3.11 showcases the meaning extracted through these metrics on a bounding box regression scenario.

Precision now signifies the ratio of intersection in between the ground truth box and the predicted box over the area of the detected box. On the other hand, **recall** represents the ratio of intersection over the object's box. These are important metrics that ensure that the accuracy obtained from the overlap between the boxes is not affected by their difference in size. For example, if the detected box is inside the object's box, the precision will be really high. However, through recall, the adjustments to be made can be deduced for it represents the percentage of intersection compared to the ground truth size. On the opposite situation, the metric's values would be reversed and likewise, the model's accuracy can still be correctly analyzed.

Summing up both precision and recall, the f1 score is an harmonic mean of both values. Its formula is described through equation 3.1. Giving a concise overview of the recall and precision values, the f1 score can describe the general relationship in between both behaviours. However, its value is unable to give the indications as to what might be happening on the model more precisely. For this reason, most experiments express recall and precision instead of a standalone f1 score.

$$f_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{3.1}$$

Some models use the metric generally mentioned as mAP, namely mean average precision. This is the averaged value of the already fore-mentioned precision.

3.2.6 Intersection of Union

In order to evaluate the quality of detected boxes through the model, the intersection of union, IoU, is made use of. This metrics represents the percentage of intersection between the ground truth and detected box over the union of the areas of both boxes. Figure 3.12 describes this relationship visually. This metric sums up the relationships delivered through the respective precision and recall. Being the most widely used metric for bounding box prediction, in studies of location prediction it is indispensable.

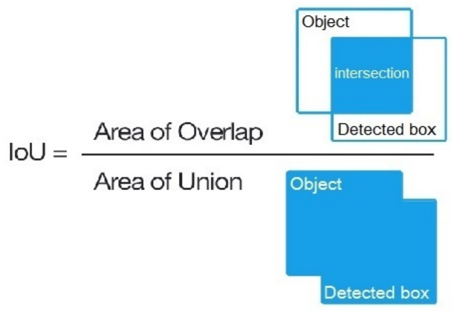


Figure 3.12: Intersection of union for bounding box regression. Adapted from [Ale18].

3.3 Summary

As an input process for object detectors, sensory information is an invaluable piece of the pipeline of autonomous driving. From a panoply of possible hardware used on autonomous vehicles, the sensors in discussion on this work are solely the RGB camera and the LiDAR. Camera and LiDAR represent data in 2D and 3D respectively. Their methods of operation diverge, however, they can leverage information that, even alone can enable object detection on the road.

Through the documentation of the shortcomings of each sensor, it is expected to create a system more sustainable in real world scenarios. Furthermore, understanding which representations each sensor can give is the key to more effectively explore the capabilities of each one.

Further than sensory perception, this chapter reflects on the issue of object detection as a specific problematic inserted in autonomous driving scenarios. On that topic, the objectives and limitations surrounding the subject are discussed as well as some constraints that also affect the end results. Autonomous driving is being pushed forward even with all the unattended concerns and the faults in technology. Leveraging the weaknesses and strengths is the path to follow.

Finally, metrics are discussed and thoroughly explained in order to give the reader a grasp of how the evaluation processes evolved.

Chapter 4

Envisioned Fusion of Camera and LiDAR Based Object Detection

Leveraging sensory information including categories such as cameras and LiDAR sensors is a complex issue that has been the target of many research works. Even though the work at hand does not focus specifically on the intricacies that come with this process, a vision of a working pipeline to combine information extracted from both sensors is proposed.

In order to provide a methodology to fulfill the pipeline, later on, a bundle of experiments performed with YOLO on camera images are presented throughout the sections to be found below. Experimentation was carried out in a matter of increasing conceptual complexity.

4.1 Pipeline

As every object detection approach relies on great compilations of data, it is of interest to explore the intricacies of the datasets that were found to be of interest. Further than that, a pipeline is envisioned which has the objective of joining information extracted from cameras and LiDAR. Its full description is then featured.

4.1.1 Dataset Statistical Analysis

Some statistics on the dataset's distribution are drawn in order to make some informed conclusions when the metric analysis is carried out.

As one can denote from 4.1, the raw dataset has an abundance of *Car* labels which is many times larger than any other class. From the start, this showcases class imbalance which will make it harder for the model to learn to correctly classify other classes. The most notable imbalance though, is the lack of *person (sitting)* labels. This eventually leads to the hardened task of learning to classify this class, making its average IoU very low.

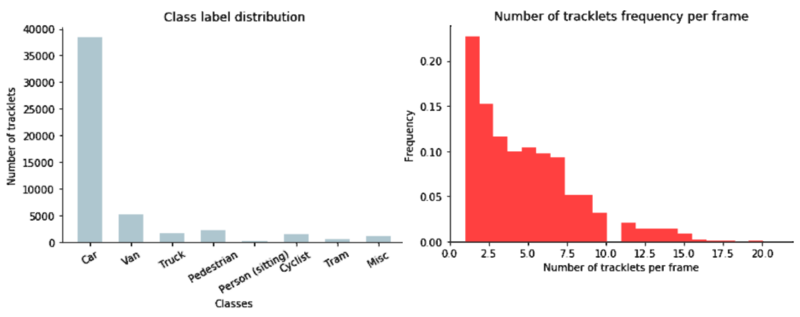


Figure 4.1: Statistics on the KITTI raw data.

Moreover, the distribution of tracklets by frame is rather varied. Having no more than 20 at any time, most of the frames on the raw dataset showcase a number of tracklets in between one and five. Larger number of tracklets by frame starts to get more rare as the number increases. Frames with more than 12 tracklets are hard examples. With that many objects on the scene, the task of distinguishing in between them is much more difficult.

All in all, the raw dataset poses many challenges to object detection and classification. Its use is directly related to the experimental analysis of the 2D detection and classification algorithm. On this dataset, it becomes interesting to observe the impact of changes on a network's hyper parameters.

For effects of training and testing on this bundle, some conceptual choices were made. Since this collection encompasses all frames taken during the trips made with the test vehicle, several consecutive images are very similar with small changes. Due to this fact, a validation dataset was drawn from 355 frames. The validation set always contains frames from different drives since inside the same drive the variation of data is very small. Even though the validation set is small compared to the number of existent frames, the data extracted is very significant since it eliminates any kind of repetition and similarity in between most of the frames from training. Through this process, the validation results presented are more meaningful.

On the other hand, the object dataset, even if still figuring plenty more *Car* labels, the ratio of unbalance is more smooth. Even so, the classes *Pedestrian* and *Cyclist* are harder to localize and classify due to their small, similar size and shape. Figure 4.2 showcases the relationships between labels and tracklets through frames.

The distribution of tracklets shows a better flow, having almost a logarithmic descent through the increase of the number of objects by frame. Hard examples such as containing around 20 objects are very rare to find. Overall, the frequency is more distributed around 1 and 5 tracklets.

As it can be denoted, this dataset is a lot more evened out and training operations with it have proved to be much more stable. Accuracy also receives a boost as the class imbalance is toned down through including a more balanced ratio of all labeled objects. On this case, it is possible to use the popular split of 60/40% in order to create the training and validation sets.

This dataset is used in all benchmark comparison attempts and even though it is possible to extract labels on all eight classes, only the three main ones are extracted and validated. On the 3D

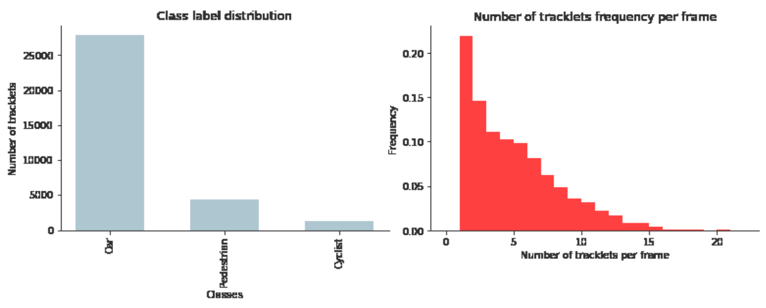


Figure 4.2: Statistics on the KITTI object dataset.

methodologies analysis, this dataset is used exclusively for training in order to ascertain the best possible outcome.

4.1.2 Envisioned Fusion of Models

The issue of joining together information obtained from different sensors is a great topic of research that is still being explored. A pipeline is proposed for the leveraging of both camera and LiDAR based extractions in order to achieve a more accurate three dimensional representation of the scenario with all objects detected.

The experimental process follows then a pipeline including all the interaction in between sensory data detection. Figure 4.3 describes the connection in between sensors that is ideally visioned. The pipeline starts with a RGB camera image of the scenario. This image is given as input to the 2D detection model, YOLO, and it outputs the bounding boxes for all objects that it found alongside with their respective class. These boxes then serve as a selective tool for the 3D workspace. The boxes are projected into the LiDAR plane and a frustum is then removed by object (guiding lines in red over the LiDAR pointcloud). The removed frustum areas are used as input to the HGNet, the BEV LiDAR object location network proposed. On the top view, detection is carried out in each individual frustum. Ideally, the extracted information would be compiled in order to obtain the 3D bounding boxes. Through the use of the height coordinate from 2D bounding boxes plus the oriented top view box from LiDAR, the 3D bounding boxes are naturally derived. Naturally following this idealized pipeline, the work focuses on exploring techniques to more effectively reach this flow.

Classification is only explored on the two dimensional space and through the YOLO model applied. This would pose as a limitation that, in future work, would be solvable by using a mixed classification model that leveraged 2D and 3D classification in order to produce the most trustable outputs. However, the main objective at hand is to explore spatial features and spatial representations for images and LiDAR scans that may aid in object localization.

For such reasons already stated, certain parts of the pipeline were prioritized. The final concatenation of the LiDAR bird's eye view detection and the camera's bounding boxes is quite intuitive. In fact, little processing would be needed when the desired information is already existent. As such, the natural directions of focused exploration were the two dimensional bounding box

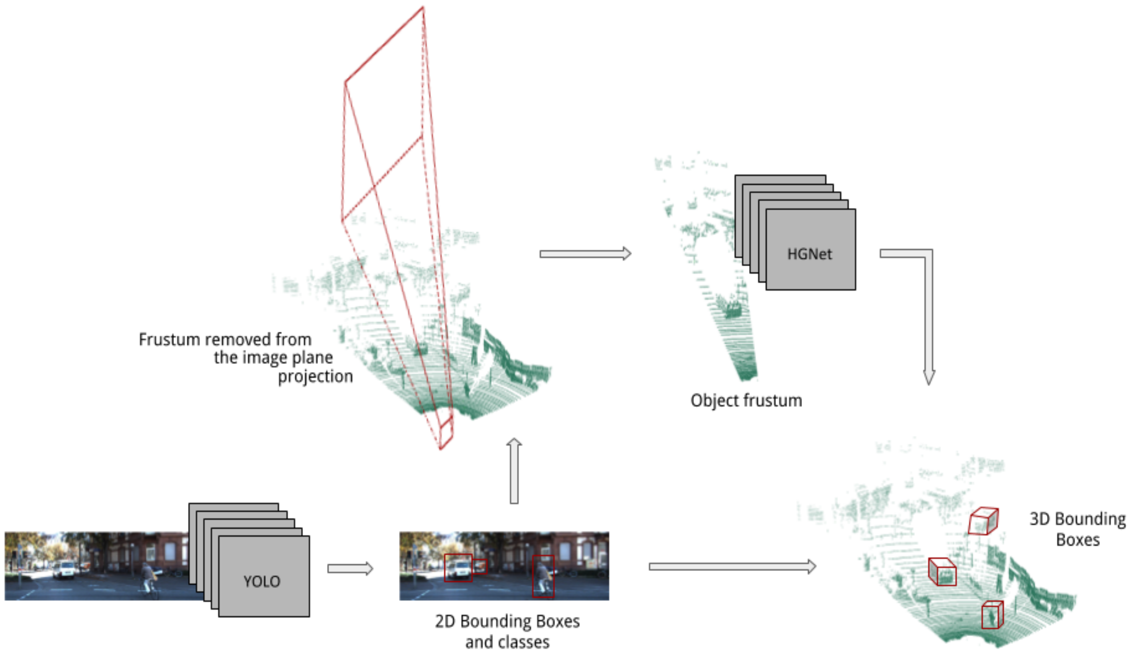

Frustum removed from the image plane projection
HGNet
Object frustum
3D Bounding Boxes
YOLO
2D Bounding Boxes and classes

Figure 4.3: Pipeline for the experimental process with sensory data.

prediction and the creation of an effective bird's eye view LiDAR representation alongside with methodologies to extract features from such ensemble.

4.2 Camera Based Object Detection Experimentation

Camera detection was the first step to be developed according to the pipeline already delineated on the previous section. As far as two dimensional object detection goes, the final process was to use a standard, general purpose detector and classifier. As the pipeline includes several steps and 2D detection serves the bigger purpose of diminishing the 3D search space, this model had to be extremely fast. In fact, it was important for it to be more fast than accurate.

Judging from these required characteristics, the most intuitive choice would be to use the *You Only Look Once Model*. As far as public state-of-the-art research goes and counting on the newest update, YOLO is the fastest model at giving detection output. Thus its relevance to this study case.

A bundle of experiences were performed using the two different dataset setups and the YOLO model version 2 and 3. The objectives of these were diverging, however, the unified goal was to achieve the configuration that would best demonstrate a great trade off in between accuracy and speed of detection. Below are presented these experiments as well as their run conditions.

All experiments described on this work were run on a single NVIDIA Geforce GTX 1080 Ti with 11GB of dedicated memory. Libraries CUDA and CudNN are used on version 9.1 and 7.1 respectively for 2D and version 8.0 and 6.0 for 3D. Generally, models were trained until they stabilized and reached a convergence stage. The weights chosen for each result comparison are those at the best early stopping point in order to avoid overfitting. Loss values displayed are taken

Table 4.1: Network resolution variation results on the raw dataset.

Network	Resolution	Loss	mAP	IOU	Val time
YOLOv2	416x416	0.30	56.06	55.05%	**4 s**
YOLOv3	416x416	0.9	72.58	65.72%	8 s
YOLOv2	256x832	0.28	69.91	59.27%	**4 s**
YOLOv3	256x832	0.43	**75.51**	**73.12%**	9 s

from an average around the early stopping point. Val time corresponds to the time that takes to validate performance on the whole testing dataset (which means detection time for 355 images on the raw dataset and detection time for 2992 images on the object dataset). The threshold used for custom mAP calculations on two dimensional models is 0.25. Training thresh equals 0.6 for the YOLO algorithm.

Experiments follow an incremental evolution pattern. In fact, if a parameter is found to be better than the others in a given experiment, in following experiences these best performing configurations are kept, varying only the tweaks in study.

4.2.1 Network Resolution

A study that was found to be interesting was the variation of the network resolution. Since the input images have a rectangular resolution (width larger than height by approximately 3.6 times), when training was deployed on squared resolutions, the end results seemed to be rather ineffective. For a squared resolution test, the $416x416$ size was chosen. Since resolutions over 832 are not advisable, the testing rectangular resolution $256x832$ was used. This is approximately a factor of 1.5 (closer resolution multiple of 32 as expected by the model) times smaller than the original dataset image size. Outcomes are layed out on table 4.1.

4.2.2 Anchors

YOLO comes with default anchors. However, for each dataset, using the specified $k-$means approach for each version it is possible to generate new anchors that are supposedly more appropriate for the training in question. Training was carried out for both versions with the default priors and with calculated ones. The results can be observed in Table 4.2.

Table 4.2: Anchor study on the raw dataset.

Network	Anchors	Loss	mAP	IOU	Val time
YOLOv2	standard	0.28	69.91	59.27%	**4 s**
YOLOv3	standard	0.43	**75.51**	**73.12%**	9s
YOLOv2	generated	0.3	68.59	59.71%	**4 s**
YOLOv3	generated	0.56	73.48	69.56%	9 s

Table 4.3: Random resolution training and its effects on the raw dataset.

Network	Random	Resolution	mAP	IOU	Val time
YOLOv2	0	256x832	69.91	59.27%	**4 s**
YOLOv2	1	416x416	61.05	55.59%	**4 s**
YOLOv2	1	256x832	22.21	54.27%	**4 s**
YOLOv3	0	256x832	**75.51**	**73.12%**	9 s
YOLOv3	1	416x416	72.58	65.72%	8 s
YOLOv3	1	256x832	11.42	50.13%	9 s

4.2.3 Random

One of the new introductions made by the YOLOv2 paper [RF17] was the enabling of the model to switch its own resolution at random in order to enable the model to generalize better for other input dimensions. Tests with this configuration were also carried out. The comparison is made between random training runs, starting either with a squared or a rectangular resolution for both models and the achieved results are presented on table 4.3.

4.2.4 Grayscale

Having trained the model on RGB images and using the exact same validation set, however this time with grayscale left pictures, the model was validated. YOLO uses saturation, hue changes and other techniques referred on previous sections for data augmentation. This experiment shows how this technique can help the model generalize better to other environments. The results were quite positive and are described in Table 4.4. Figure 4.4 shows a detection with the model on a painting, as an example for other kinds of generalization.

4.2.5 Version Comparison Analysis

Wrapping up all the experiences previously exposed, the best configurations from each version are chosen in order to directly compare how both models wage on the raw dataset. On this dataset, the best configuration found for both models does not use random training but a rectangular resolution and the standard anchors. Table 4.5 shows a strong improvement on mean average precision of around 5 points and an increase for intersection of union of over 13 percent which are incredibly substantial results for YOLO version 3 over its predecessor.

From the bundle of experiments carried out, it is possible to conclude that, overall, the YOLO model has improved considerably from its predecessor version 2 to its newest ensemble, version 3. Even so, there is still a visible trade-off in the newest update such as the loss of detection

Table 4.4: Grayscale evaluation results on the raw grayscale dataset.

Network	mAP	IOU	Val time
YOLOv2	63.79	57.59%	**5 s**
YOLOv3	**71.45**	**70.27%**	8 s

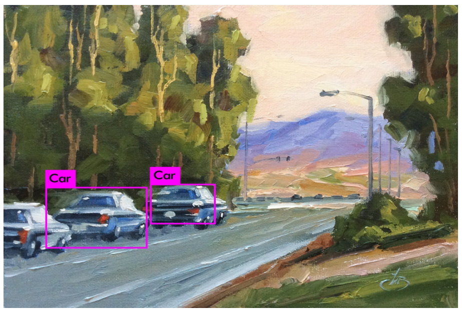


Figure 4.4: YOLO version 3 detection outcome on a painting of cars.

speed in detriment of accuracy, which is obvious in all previous experiments. In most cases, it has approximately doubled the detection time for 355 images, properly observable in Table 4.5 through the column *Val time*. There certainly is a trade-off between accuracy and speed which may influence the applications of this model in certain real-world situations. However, it is still the fastest model known publicly, producing detection in around 25 ms per image on the newest update at a large resolution (of 832 in one parameter). This behaviour was more than expected as the number of layers was more than doubled from Darknet-19 to Darknet-53.

As for resolution, the YOLO model shows different behaviours in version 2 and in the newest update. It seems reasonable to conclude that keeping the aspect ratio of the original training images is the best approach regardless of the model being used. However, it is also obvious to conclude that YOLO version 2 was much more sensible to the object's original ratio. This is proved by the astounding improvement in mAP of 13.85 points gained by using a rectangular resolution as seen in Table 4.1. Meanwhile, the improvement in version 3 is only of around 3 points. This may be due to the use of connection of the different scales and feature maps that make it more invariable to natural scales on the newest update. The lack of sensibility in between resolutions is certainly a good aspect as it may increase the capacity of the model to generalize better in new conditions.

When it comes to the prior generation, the results seem a bit puzzling. Even if the standard anchors have been properly tuned on larger datasets such as COCO [LMB+14] and PASCAL VOC [EEV+14] with a large abundance of classes and cases, it is still surprising to find that the standard anchors produce slightly better end results for both models. Nevertheless, this is good news as it means that YOLO can be adapted to new datasets with minimal pre-processing overhead. Table 4.2 shows small differences between using standard and generated anchors, having only an increase around 1-2 points on mean average precision. Even though slightly, the use of generated anchors over the standard ones for version 3 seems to produce worse results compared to the behaviour of version 2.

Table 4.5: Comparison of the best configurations laid out by version.

Network	Loss	mAP	IOU	Val time
YOLOv2	0.28	69.91	59.27%	**4 s**
YOLOv3	0.43	**75.51**	**73.12%**	9 s

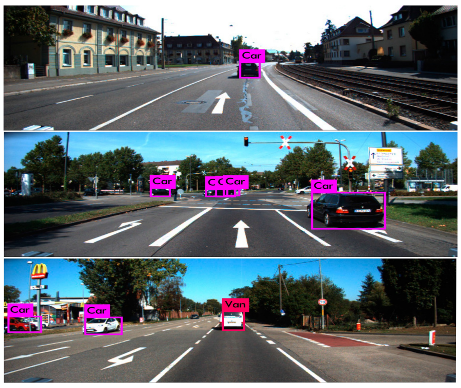


Figure 4.5: YOLO version 3 on the KITTI dataset for lane detection [FKG13].

The possibility to variate resolution during training was also tested and yielded interesting results. The use of random for training with a starting squared resolution gives good results even though the testing images are rectangular in both version 2 and 3. Even so, training for a rectangular resolution specifically proves to be the best approach, assumption proved by the results showcased in Table 4.3.

Moreover, the use of random training from a starting rectangular resolution produces rather poor results on both metrics even if more heavily on average precision. This seems to be a limitation in the classifier. Both versions of the model have the same kind of behaviour towards this parameter. From Table 4.3 one can denote that having a direct rectangular resolution produces superior outcomes compared to having a randomly sized training process which consumes more gpu resources and takes around twice the time to train. When the ratio of the images is known beforehand, it is preferable to keep the network's resolution parallel to that of the original aspect whilst still keeping it under 832 for width or height since larger resolutions have a directly proportional decrease in detection speed that would nullify the objective of trading off speed over accuracy. This was also tested in extra experiments.

YOLO is able to generalize to different resolutions, environments, lightning conditions and even paintings. On the grayscale experience, with results showcased in Table 4.4, it is only noticeable a little drop in mean average precision compared to the colour validation set. This is an invaluable quality for an autonomous driving situation. On the road, many unplanned, new scenarios arise and the model needs to be elastic enough to respond to all of them with the degree of safety and confidence as the cases it was trained for. On this aspect, YOLO maintains and in some aspects even increments on its generalization value. Table 4.4 depicts less loss in average precision in the newest version than in its predecessor whilst comparing these with the best obtained on the RGB set.

For all the reasons listed, YOLO version 3 was the final choice made for a 2D detector. Since its speed kept up with the requirements whilst its accuracy had an incredible boost, this seemed to be the most sensible decision. On table 4.6 are the final, best results obtained with YOLO version 3 on the raw dataset figuring the best found hyper parameters.

Table 4.6: Best performing hyper parameters.

Network	Resolution	mAP	IOU	Val time
YOLOv3	256x832	82.31	74.70%	315 s (for train dataset with 12131 images)

4.3 Benchmark Environment

Opposing the previously exposed environment, the experiments to follow have the intention of providing an overview of the model's performance on the benchmark target classes and methodologies.

4.3.1 Training on Raw Dataset

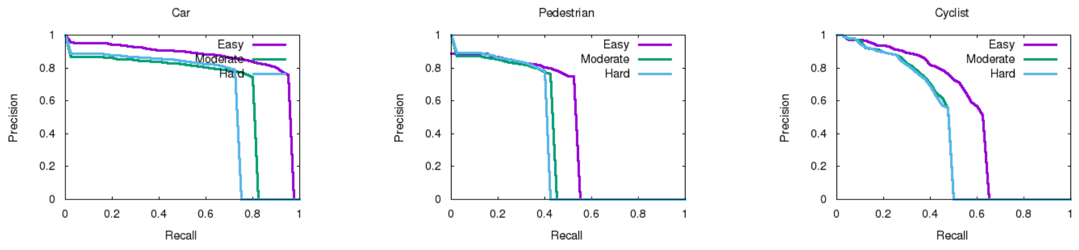
On the following experiment, the training weights and configuration for the raw dataset are used for evaluation on the object dataset. Described on table 4.7 are the outcomes. As it can be denoted, specially on the *Easy* category, for the label **Car**, the accuracy is maintained. However, the model generalizes wrongly to other classes such as pedestrians and cyclists. This outcome is entirely expected. Since the model was trained on the unbalanced dataset, the classification accuracy of any class beyond *Car*, such as *Pedestrian* and *Cyclist*, specifically on this case, suffers greatly. Graphics 4.6 showcases the relationship between the two used metrics recall and precision. From the relationships between recall and precision, it becomes evident that the model is able to keep its precision high, however, the recall reaches considerably lower values specially on the *Pedestrian* and *Cyclist* classes. This means that the model has low rates of false positives. On the contrary, it features a high value of false negatives. In fact, the model fails at finding the location of many pedestrians and cyclists on the validation set.

4.3.2 Training on Object Dataset

Incrementally, from the previous experience, the same setting is repeated, however, this time using the configuration and weights from training on the object dataset. Using a train/val split of 60/40% respectively, for the splitting of the object dataset, the results are laid out over iterations, classes and difficulty levels. The difference in between the use of the datasets becomes obvious through this experiment. Even though the raw dataset contains most of the frames on the object dataset, the more balanced training enables the model to reach great precision and specifically, recall rates which, on the end results, brings a staggering impact over all classes.

Table 4.7: Average precision by class for the raw training dataset experiment.

Class	Easy	Moderate	Hard
Car	81.88	68.80	63.18
Pedestrian	45.68	39.28	39.77
Cyclist	53.74	39.94	39.49



Car Pedestrian Cyclist

Figure 4.6: Precision recall curves for the raw training dataset experiment over the three detected classes.

The results showcased by the model on table 4.8 are outstanding, reaching a maximum accuracy of 98% on the easy category for Cars, 85% for Pedestrians and 86% for Cyclists on the validation set. Even on the hard category, which, as explained on the KITTI dataset web page, exists only as a reference, the maximum values achieved are impressive. As stated, around 2 % of the hard boxes are not recognized by even the human eye.

Opposing the previous experimental environment, the model still performs accordingly well for pedestrians and cyclists. On this case, it was expected that YOLO would bring out less accurate results since it has been appointed that the model has difficulties with smaller boxes. Even so, the results achieved on this experiment showcase that there has been an evolution of the model on this specific domain.

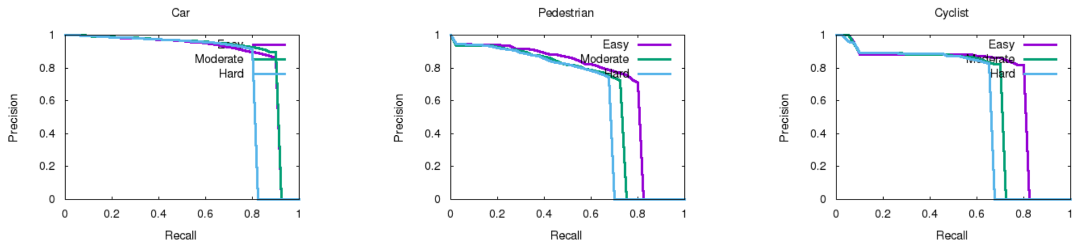
Benchmark positions are ascertained through the *Moderate* level of difficulty. As such, on this category, cars reach a maximum mAP value of 89.9%, whilst for pedestrians and cyclists the value circles around 77%.

Overall, the 2D detection results seem to have a great balance of accuracy and detection speed. Specifically on the classes *Pedestrian* and *Cyclist*, the model showcases an accuracy fitting of the first 5 models on the benchmark. All the while maintaining a much higher detection speed in general.

Figure 4.7 showcases the recall precision curves for one iteration from table 4.8. The difference can be denoted from the previous featured curves (figure 4.6). On all three classes precision

Table 4.8: Comparison of results obtained on the object dataset using the YOLO version 3.

Iteration	Car			Pedestrian			Cyclist		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
15400	**98**	**89.9**	80.99	76.27	67.37	66.86	85.54	66.6	66.27
16000	97.06	89.48	80.62	77.18	68.2	67.76	**86.41**	68.43	67.85
16600	89.52	89.76	**89.3**	76.04	67.26	66.83	74.59	64.45	64.67
17000	89.5	89.47	80.69	78.3	**76.91**	69.02	85.32	**76.59**	**68.39**
17500	97.78	89.7	80.83	**85.09**	76.19	68.26	77.81	67.85	67.72
17600	89.81	89.65	80.85	78.57	69.64	**69.1**	76.28	67.11	67.08



Car Pedestrian Cyclist

Figure 4.7: Precision recall curves for the object training dataset experiment over the three detected classes.

starts at 1.0 and as recall gets higher, the precision stays almost squared on the higher levels. Even though, it is still visible a clear gap in between the classes Car and Pedestrian/Cyclist. The reason that justifies this behaviour is not only based on the smaller size presented by these objects but on the under sampling of these classes comparatively to the Car class as seen on figure 4.2.

For more insight into the training evolution of the YOLO model on the object dataset, refer to annex A.3.

4.3.3 State of the Art Comparative Analysis

Tables 4.9, 4.10 and 4.11 showcase the results that are found on the benchmark environment for all classes and categories. As ranking is awarded through the *Moderate* difficulty, the positions of the models are respectively shown. Furthermore, the bold values represent the best performing ones for each category.

One model dominates the first ranking for all classes. The iDST-VC/iDST-VT is an anonymous submission with no information available besides the results showcased on the benchmark. However, for all its rankings, the registered runtime is quite high, reaching even the huge value of 4 seconds on the *Car* category. It seems that the model can not be applied on autonomous driving scenarios as it is, being quite far from the desirable detection speed.

The leading accuracy values on the *Car* class reach the incredible value of 90.55 on the *Moderate* category whilst, for *Easy* and *Hard*, mAP amounts to 97.25 and 87.44, respectively, as seen on table 4.9. Out of the published methods, the RRC [RCL+17] and the Deep MANTA [CCR+17] dominate the leaderboard on the Car class.

On the *Pedestrian* class high scores are comparatively smaller, refer to table 4.10. The F-PointNet [QLW+17] is one of the few published methodologies on the top of this specific board.

Finally, the *Cyclist* class, even if not featuring mAP values as high as the *Pedestrian* class, has a more distributed gap in between the leading models. Once more, the RRC [RCL+17] model proves its value with great results.

In order to be able to compare YOLO with the current state-of-the-art models that have the best ranking on the benchmark, a private submission was made. The results that were obtained are

Table 4.9: KITTI benchmark top models for each category on the *Car* class.

Car				
Ranking	Model	Moderate	Easy	Hard
1	iDST-VC	**90.55**	90.88	81.04
6	RRC [RCL+17]	90.22	90.61	**87.44**
11	Deep MANTA [CCR+17]	90.03	**97.25**	80.62
16	SG	89.89	90.51	80.67

showcased through table 4.12. As it can be denoted, YOLO has maintained a mean average precision much below what would be expected of it. On the *Car* class, the drop in average precision is not as substantial as the other classes. Even so, it is still under the mark that would be expected on the autonomous driving scenario.

On classes *Pedestrian* and *Cyclist*, the average precision seems to be severely under the expected result pool. Specifically on the ranking category, the *Moderate*, the average precision is around 30% which is rather low compared to the top models discussed previously.

Through the analysis of table 4.12, it seems possible to conclude that YOLO has a bigger tendency to classify *Easy* examples substantially better, whilst the gap towards other difficulty levels enlarges. Once more, its difficulty in identifying smaller objects becomes evident through the large gap in precision found in between classes *Car* and *Pedestrian*, *Cyclist*.

There is not enough data to make informed conclusions on the speed of the benchmark methodologies. Even so, from the published models that feature on these top positions, no paper has proved to have detection speed values even close to that of YOLO. For this reason, YOLO still maintains its great advantage on the real time application scenario. Even so, the precision that was reached can not ever be transported to the real world. For this reason, the model showcases not to be indicated for this specific environment.

One thing that can be denoted on the benchmark is that it is rather hard for a model to hold high ranking positions on all classes at the same time. In fact, the testing dataset in general seems to represent more challenging situations than that of the training database. This is also a factor to be taken into consideration when analysing these models.

Specifically, the benchmark only counts results for the AP calculation if the bounding boxes overlap is greater than 70% for cars and 50% for both pedestrians and cyclists. Redmon et al. [RDGF16] have demonstrated that most errors that occur on the YOLO model are based on loca-

Table 4.10: KITTI benchmark top models for each category on the *Pedestrian* class.

Pedestrian				
Ranking	Model	Moderate	Easy	Hard
1	iDST-VC	**80.90**	**88.13**	74.08
3	F-PointNet [QLW+17]	77.25	87.81	**74.46**
5	VCTNet	75.88	84.03	71.60

Table 4.11: KITTI benchmark top models for each category on the *Cyclist* class.

Cyclist				
Ranking	Model	Moderate	Easy	Hard
1	iDST-VT	**78.21**	**86.06**	**69.47**
4	RRC [RCL+17]	76.47	84.96	65.46

tion mistakes. It is observable through testing on the KITTI dataset that the objects are identified correctly, however, the bounding boxes miss a part of the objects more often that not and are sometimes wrongly centered. This would lead the results to not having the overlap that is admissible by the benchmark and thus the identification of these objects would not be counted towards the average precision metric. It is believed that this fact is one of the major factors contributing to the disappointing results obtained.

Furthermore, the fact that the testing dataset is populated with many samples containing several small objects grouped such as groups of pedestrians, cyclists or even bicycles alone. These are cases in which YOLO starts to show weak results for its difficulty with smaller objects is denoted even more.

4.4 Summary

Building up from the dataset's fundamental statistics, some general considerations on the distribution of the samples are drawn. The raw dataset provides an unordered bundle of frames whilst the object dataset is specifically built for detection training on only three main classes - *Car*, *Pedestrian*, *Cyclist*.

The first step taken on the implementation phase was the envisioning of a pipeline that would jointly leverage information from 2D and 3D object detection methodologies. Even if not entirely implemented, this setup is described fully, delineating the prioritized tasks.

Moreover, the experiments made with camera images and YOLO are featured as an incremental process. Playing with the network's hyper parameters and conceptual architectures, several configurations are tested in search for the best performing one. Besides this, the two competing versions of YOLO, the second and most recent, are compared in both quantitative and qualitative result pools. The submission to the KITTI object detection benchmark is carried out and the disappointing results are respectively discussed.

Table 4.12: YOLO evaluation on the KITTI Vision Object Detection Benchmark.

Model	Car			Pedestrian			Cyclist		
	AP_{70} Easy	AP_{70} Moderate	AP_{70} Hard	AP_{50} Easy	AP_{50} Moderate	AP_{50} Hard	AP_{50} Easy	AP_{50} Moderate	AP_{50} Hard
YOLOv3	88.11	78.80	69.68	48.08	33.07	32.53	45.03	29.94	29.73

Chapter 5

The HGNet Approach for LiDAR Based Object Segmentation

In order to aggregate and segment the areas of objects to be found on LiDAR, a novel network is proposed that leverages the focal loss for binary classification in order to solve the proposed task.

5.1 Preprocessing

Processing pointclouds is not an easily achieved computational task. In order to ascertain that the task of finding the objects in 3D LiDAR would be able to achieve the desired detection speeds, some preprocessing steps are carried out before any feeding to a neural network.

5.1.1 Frustum

The biggest hurdle to overcome when searching 3D space is to find an effective methodology of search space reduction. Since pointclouds are rather extensive, processing them all at once is quite a challenging process for current computers. Even more, these pointclouds usually include huge gaps of empty space spread out through three dimensional space. Processing this emptiness only increases the time needed to analyze the space and is, in the end, a wasted effort of resources. As such, it becomes urgent to define a preprocessing step that enables the search space to be reduced and at the same time to minimize the empty sides of the pointcloud.

The chosen approach is based on a recent model proposed by Charles Qi et al. [QLW+17]. On the mentioned paper, for each object found in two dimensions, a frustum is removed from the pointcloud. From the projection of the 2D bounding box over the frustum, a slice of three dimensional points is removed. Through this process, it is guaranteed that each frustum contains one object on the minimum.

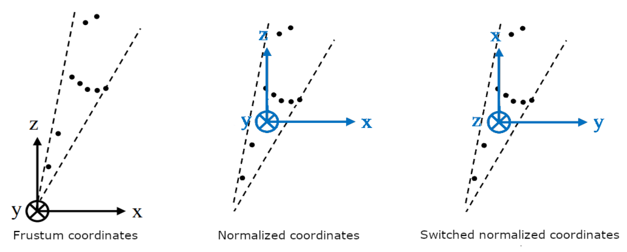


Figure 5.1: Frustum and normalized coordinates axis change. Adapted from [QLW+17].

The frustum projection is decided through the use of the calibration matrix available for the dataset in order to transform from the camera axis into the LiDAR coordinates. Points inside the space limited by the 2D bounding box are then extracted and compose the object frustum.

The coordinate system of the frustum is represented on figure 5.1 by the most left aligned coordinate system. The slice is removed in a certain angle from the orthogonal natural LiDAR coordinates.

5.1.2 Normalization

Since frustum angles can be quite random depending on the structural shape of the pointcloud and the object's location, the resulting pointclouds may be quite different even if the same object is present in two removed frustums in the same orientation. In order to facilitate the model's task of learning through repetition and connection of similarities a normalization step is performed after the frustum removal. This normalization makes the frustum centroid the new origin axis which also contributes to accelerate the end model's biased behaviour towards frustum rotation. This is a design choice in order to be able to extract box orientation directly through the training of a segmentation three dimensional space angle-sensitive model. Since the rotation of the boxes is also affected by the frustum angle, this provides a platform for the model to learn more spatial intuitive orientations or tendencies towards directions.

Through normalization, the axis is shifted to the centroid which in turn becomes the new reference, seen through figure 5.1 on the center plane. After this step is finalized, the coordinate system is switched so as to have x reflect depth and y the width through the frustum.

5.2 Representations

Since LiDAR scans feature three dimensional features, it is important to dwell on a representational method so that the points can be fed to a model. Even if it seems to be possible to work with all three dimensions at the same time, as proposed by Charles Qi et al. [QLW+17], the use for leveraging of three dimensional features is deemed useless since the two dimensional model can give the classification and the respective height coordinate. As such, in order to better allow for fusion with the 2D based modeled output, a representational method is proposed and explored so as to extract the top view oriented boxes.

5.2.1 The Bird's Eye View

Bird's eye view corresponds to the upper view of the pointcloud. Detection of 2 dimensional bounding boxes on LiDAR seen from this perspective has been more studied than natural processing of 3 dimensional points. Since representations in bird's eye view are more intuitive to the way humans process and deal with geometrical information, more advancements were made on the topic. This work dwells on LIDAR detection methods for this specific perspective due to the fact that it is advantageous for the currently designed pipeline.

This view is constructed through the elimination of the height coordinate so as to be a representation of what would look like if the pointcloud was being seen from above (a bird's point of view). A plane on *xy* is then constructed and the third axis is shown as the flat difference through heights.

5.2.2 The Height Grid

A new representation is drawn from LiDAR bird's eye view so that the tasks carried on by the networks are made easier. From each removed frustum, the points are transformed into a bi-dimensional grid that saves a variable height value found from the points aligned within the cell in question. When points fall under the same pixel, a sampling process in ensued. Opposing other typical sampling procedures that only select a bundle of points at random, the use of this grid enables the sampling of points that have a high overlapping percentage guaranteed that the grid dimensions are high enough. Grid dimensions then represent a specified sample rate, that may be different or not in both axis. The resulting grids can be interpreted as images, outputting a representation close to that of a color map representing height. Figure 5.2 shows an example of a height grid of dimension $(160, 128)$ built from a frustum with an object of label *Car*. The representation on the right showcases the pixels that belong to the object in question. The remainder of points in the frustum are either ground points or noise generated by other kinds of objects.

This representation is based on Yu et al. own's elevation grid [YWH+17]. On their version, the resulting images are of the RGB type, having the three channels be simulated through the maximum, medium and minimum height value at each cell's location. Through the curiosity to explore the efficiency of this method on the grayscale format, the height grid was then thought of.

The size of this grid is further explored on the experiments to follow. However, it is intuitive to understand that the sampling rate has an effect on the final representation obtained. The smaller are the dimensions the more points will fall under the same pixel and precise location will be lost, leading to errors on models that would use this representation non wisely. Even so, having a size too big will not be advantageous either since the grid will become too expensive to create or process, leading to close to no sampling. The choice of the grid size is an important process for its advantageous use. As such, it must be carefully dwelt on.

When these grids are extracted, they are represented as grayscale images with the dimensions of the grid and with values ranging in height that will showcase different shades of grey on the final extraction.

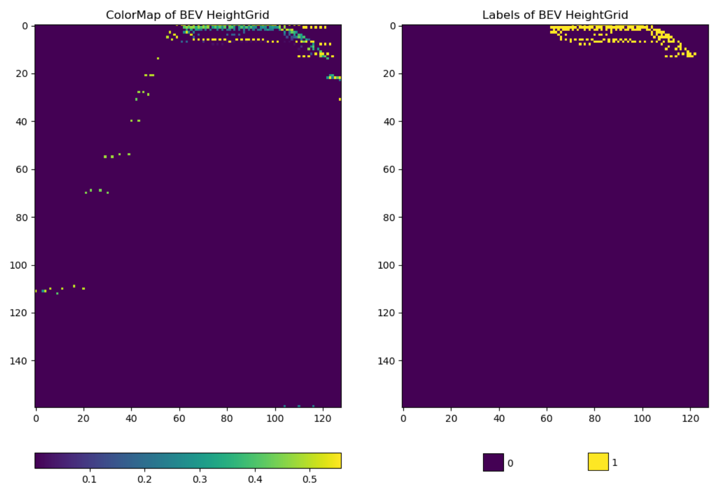


Figure 5.2: Color map of a height grid in a frustum with the respective pixels belonging to the object.

5.3 Eigenvector Based Ground Segmentation

The removal of the ground points is also studied on full, 3D front view coordinate based scans. Even though most authors remove the ground plane through the fitting of a plane to the lower points, such as [KML+17], the current work studies the possibility of using the eigenvectors and eigenvalues of the pointcloud for the same objective. The effect of the three largest eigenvalued eigenvectors is then analyzed on front view pointclouds using the raw, non normalized points. The observations presented below were found through the study of several pointclouds from the raw dataset.

First eigenvector Represents the first axis of variation of the 3D points. Usually leads forward.

Second eigenvector Second most prominent direction of variation. Important when the road is perpendicular to the vehicle's positional orientation.

Third eigenvector Manifestation of the height distribution.

The study of eigenvectors obtained from the pointcloud can be an effective measure for extracting spatial, coordinate based features. Furthermore, the direction of the road can be efficiently and accurately deduced. This was tested through an eigenvalue priority ordering of the eigenvectors. If the second eigenvector has an eigenvalue similar to that of the first, then the road follows the direction indicated by that vector (on most cases perpendicular - for example in crossroads). Using this simple evaluation process, the ground plane is found rapidly.

More than this, the use of eigenvectors for pointcloud analysis can bring up many advantages for many other related tasks such as lane detection and object tracking. A further study of these properties is ensued. However, it is shown that great potential is contained on the use of these properties of the data.

5.4 Using $k-$means Clustering for Object Segmentation

In order to enable the finding of the object's location on the frustum, it is of interest to understand which points on the scenario belong to the object or not. This process, generally denominated as segmentation, was tested through machine learning and deep learning techniques.

As a base model, $k-$means clustering was tested for segmentation on the bird's eye view. Through the creation of a height grid, the filled pixels were extracted and extrapolated back to frustum coordinates. Using the samples obtained through this process, the behaviour of the clustering process was observed.

Following up to the experimental and observation process, several empirical observations were made on the effectiveness of the $k-$means clustering.

- The optimal k factor depends on the grid and object dimensions.
- The same k could not be effective for classes *Car*, *Pedestrian* and *Cyclist* simultaneously.
- The k factor to be used is not trivial to find, increasing the pre processing overhead greatly.
- Clustering is unable to extract object related spatial features.
- The features accentuated were the density distribution of points.
- Outliers were a real danger to the effectiveness of the segmentation.
- Clustering segmentation achieved bad results in terms of precision and recall due to the high number of false positives and false negatives respectively.
- End results featured a low 0.2 f1 score.

In order to filter outliers, the local outlier factor technique was experimented on. This methodology was used in an unsupervised manner. In fact, some data with examples of expected points to be considered outliers alongside with shapes of interest were generated in order to make sure that the model would consider as outliers the points that do not contribute to the formation of a shape.

Even if this method was able to remove some of the outliers, the end results were still rather weak for these types of approaches are unable to learn spatial features which are invaluable for the process of segmentation. Furthermore, machine learning methodologies are rather slow compared to the current deep learning models.

These empirical observations have been corroborated through literature. Several works have made use of the $k-$means in order to segment all kinds of data. For a fact, many works have delt with the difficulty of finding an optimal k factor for segmentation tasks. On this topic, Kalam *et al.* [KM11] have studied the possibility of optimizing the initialization process. However, even so, clustering proves to be much slower than deep learning and unable to identify several features at once. More closely related to LiDAR based segmentation, the work of Chehata *et al.* [CDB18] reaches good results with the $k-$means clustering of terrain from landscapes. However, they also realize the sensitivity to parameters showcased by the algorithm.

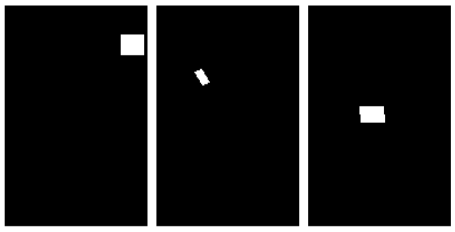


Figure 5.3: Some examples of labeled images with filled bounding boxes from the evaluation dataset.

As such, it becomes evident that the $k-$means clustering approach is not great for the segmentation of objects on LiDAR based pointclouds. Even if good results would be possibly achieved, these would be too specific to each scenario and would implicate a huge processing overhead.

5.5 The Height Grid Network

Building up from the failed experiments and running towards the desire to provide a reliable method for effective segmentation, a deep learning approach is then adopted. A network is proposed, HGNet, namely Height Grid Network, which works on the height grid representation previously exposed and combines a v shaped model with a focal binary crossentropy loss.

5.5.1 Input Preliminaries

Using the frustum analysis previously exposed, the data is then transformed into a frustum dataset in which each entry includes only the points inside each frustum. The objective for each frustum is then to analyze the location of the oriented object in bird's eye view. Having loaded the 2D detection results into a python pickle object for frustum, these are loaded through a generator which transforms each pointcloud into an height grid. The height grids are the inputs for the model and are processed in the form of an image in which each x, y pixel represents a location on the BEV LiDAR coordinates and the channels contain information on the height of the points contained on each pixel.

The channels used can be one of the motives of study for the network. The first approach was to use only the maximum height value found for each pixel. The end result from this grid has the sort of representational significance as a relief map. However, some works such as Yu *et al.* [YWH+17] have proposed using a channel organization based off of RGB, through the use of the maximum, minimum and median elevation levels. The use of only the highest coordinate is an interesting study case which minimizes the steps of preprocessing and speeds up computation. Moreover, it is possible to explore models that work on contrast imagery since the relief color map presents a similar color gradient towards the object. This affirmation is based on the fact that, generally, on a frustum, the only mass of points that present high and low height values closely are the objects of study.

The issue is digested as a binary classification problem. In fact, it is the objective that the pixels that are inside or near the object's bounding box area are classified as 1 whilst all other pixels are labeled as 0. The methodology of labeling training samples was experimented on as well. At first, labels were simply the classification of pixels that contained existent points belonging to the object. This representation is of the sort of 5.2 as seen on the right. On the other hand, the other kinds of labels explored were simply the use of an image with black background and a white bounding box drawn over the pixels that fall on it's area, figure 5.3. Whatever labeling methodology made use of, this leads to a very unbalanced binary classification issue. Most samples are of class 0 since height grids have a huge amount of empty space that is translated from the pointcloud's distribution. Using traditional accuracy methodologies, the results would be erroneous since even if the model predicted all labels as zero, it would have gotten more than 70% of the labels right. Through the observation of this behaviour, it was decided to deal with the metrics of precision, recall and f1 score. These metrics provide a more meaningful comparison basis. Moreover, this is the reason why the use of the focal loss was studied. Through its use, even if the unbalanced data is dangerous for the model, the training is able to proceed smoothly and the learning task is successfully completed as the easily labeled class does not overwhelm the gradient.

5.5.2 Architecture

The network's architecture is represented on figure 5.4. The diagram showcases a v-shaped design featuring two main paths - contraction and the followup expansion. In fact, the image is consecutively downsampled through chained two dimensional convolutions. Then, once a certain minimum is reached, the image is upsampled through each level that it went through on the contraction path back to its original input shape. The output format is then a grayscale image featuring binary classification values for each pixel.

Through the contraction and expansion paths, the activation used is the relu function except for the sigmoid on the outputting layer. Dropout is made use of with a factor of 0.5. All convolutional operations are simple and performed with a kernel size of $(3,3)$, featuring a stride of 1, in the fashion of the VGG Net [SZ14].

At each dimensional level, a chain of convolution is carried out. The process followed on each feature map dimensional level is detailed on figure 5.5. The input or the concatenation from the previous level is fed to a convolutional layer. The channels are doubled from the previous size. After some dropout, another convolutional layer is added with no impact on the shape. A concatenation is always ensued. This layer joins the current feature map with the previous max pooling directed output. Through this operation, the feature maps are enriched, making sure that the previously extracted information is not lost and fusing extraction outputs throughout all the contraction path. The dimensions of the feature map are then downsampled through the max pooling that cuts in half the size on the x and y axis. The channels then correspond to the concatenation resulting shape.

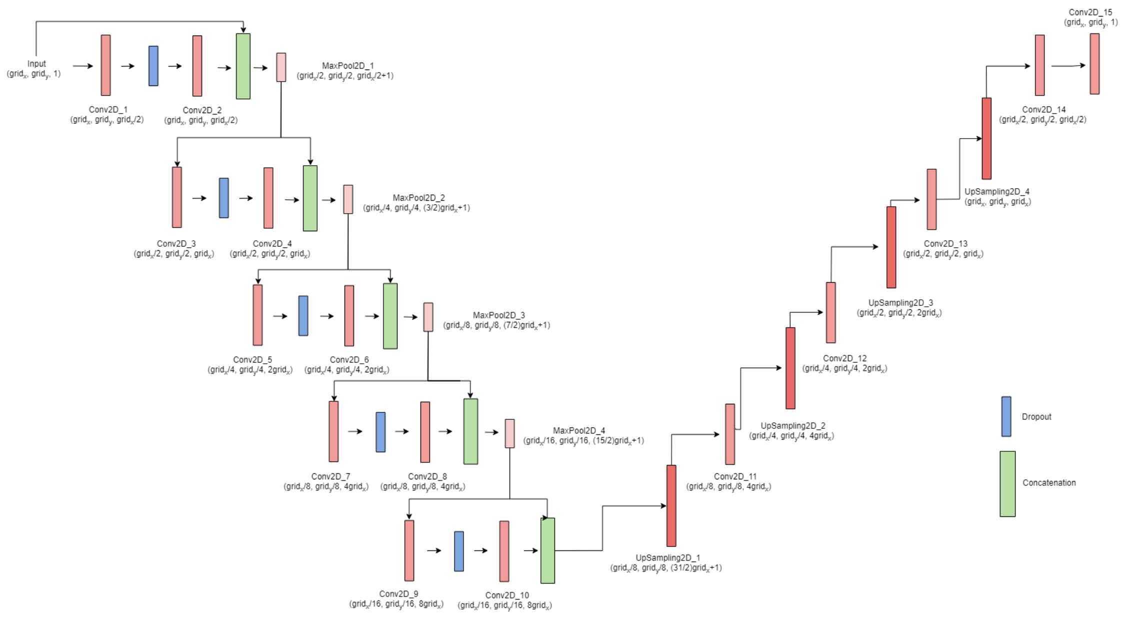

Input
$(grid_x, grid_y, 1)$
Conv2D_1
$(grid_x, grid_y, grid_x/2)$
Conv2D_2
$(grid_x, grid_y, grid_x/2)$
MaxPool2D_1
$(grid_x/2, grid_y/2, grid_x/2+1)$
Conv2D_3
$(grid_x/2, grid_y/2, grid_x)$
Conv2D_4
$(grid_x/2, grid_y/2, grid_x)$
MaxPool2D_2
$(grid_x/4, grid_y/4, (3/2)grid_x+1)$
Conv2D_5
$(grid_x/4, grid_y/4, 2grid_x)$
Conv2D_6
$(grid_x/4, grid_y/4, 2grid_x)$
MaxPool2D_3
$(grid_x/8, grid_y/8, (7/2)grid_x+1)$
Conv2D_7
$(grid_x/8, grid_y/8, 4grid_x)$
Conv2D_8
$(grid_x/8, grid_y/8, 4grid_x)$
MaxPool2D_4
$(grid_x/16, grid_y/16, (15/2)grid_x+1)$
Conv2D_9
$(grid_x/16, grid_y/16, 8grid_x)$
Conv2D_10
$(grid_x/16, grid_y/16, 8grid_x)$
UpSampling2D_1
$(grid_x/8, grid_y/8, (31/2)grid_x+1)$
Conv2D_11
$(grid_x/8, grid_y/8, 4grid_x)$
UpSampling2D_2
$(grid_x/4, grid_y/4, 4grid_x)$
Conv2D_12
$(grid_x/4, grid_y/4, 2grid_x)$
UpSampling2D_3
$(grid_x/2, grid_y/2, 2grid_x)$
Conv2D_13
$(grid_x/2, grid_y/2, grid_x)$
UpSampling2D_4
$(grid_x, grid_y, grid_x)$
Conv2D_14
$(grid_x/2, grid_y/2, grid_x/2)$
Conv2D_15
$(grid_x, grid_y, 1)$
Dropout
Concatenation

Figure 5.4: The HGNet architecture breakdown.

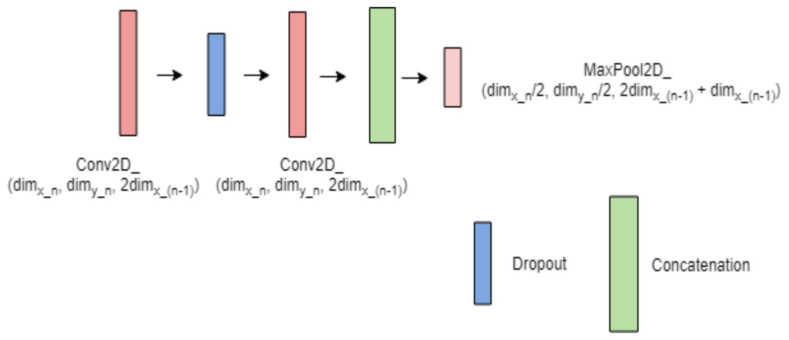

Conv2D_
$(dim_{x\_n}, dim_{y\_n}, 2dim_{x\_(n-1)})$
Conv2D_
$(dim_{x\_n}, dim_{y\_n}, 2dim_{x\_(n-1)})$
MaxPool2D_
$(dim_{x\_n}/2, dim_{y\_n}/2, 2dim_{x\_(n-1)} + dim_{x\_(n-1)})$
Dropout
Concatenation

Figure 5.5: Detail on the convolutional chain found at each feature map dimensional level of the contraction path.

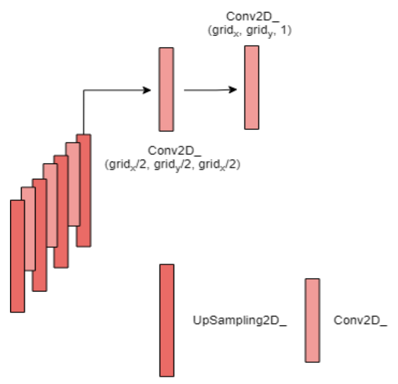


Figure 5.6: HGNet expansion path overview.

The contraction path leads to the creation of feature maps up to 16 times smaller than the original input image size. The input dimensions are kept at multiples of 16 in order to ensure that there is no loss of information through the reducing dimensionality.

The expansion path is rather simple, containing only a bundle of bi-dimensional upsampling layers that double the size of the previous feature map. Each expansion is followed by a simple convolutional layer which divides by half the channels from the previous layer. This leads to the increasing pixel dimensions across the x and y axis. At the same time, the channels suffer the opposing process of shrinking down until they hit their minimum, labeling style size, the one or zero binary classification of the pixel. Figure 5.6 showcases this process. As it can be denoted through the imagery, the final step includes two convolutional layers that simply downsize the channels back to one. No batch normalization and or dropout is carried out on the expansion path due to the possibility of losing extracted features and its proven unnecessary application.

5.5.3 Focal Loss for Binary Pixelwise Classification

Adapted directly from the focal loss proposed on [LGG+17], the loss function used for training of the HGNet is presented through equation 5.1 where pt represents the matrix of predicted labels intersected with the ground truth for each respective class. The $m \times n$ represents the number of pixels existent on each image of size (m, n) and so, all pixelwise classification outputs contribute to the final loss value. Due to this fact, the loss values tend to be quite high during training and evaluation procedures. This does not interfere with the procedures of the model as values are put in a comparative perspective.

$$L = -\sum_{i=0}^{m\times n} \alpha(1-pt_1)^{\gamma}\log(pt_1) - \sum_{i=0}^{m\times n}(1-\alpha)pt_0^{\gamma}\log(1-pt_0) \tag{5.1}$$

Like the original focal loss, this loss function includes an attenuating factor to balance the variation, the α. Moreover, the modulating factor is added with a variable γ parameter. The logarithmic expressions come from the cross entropy loss for binary classification and, on the implementation, contain an epsilon in order to avoid infinite or nan values during training and/or evaluation.

Table 5.1: Results obtained through threshold variation.

Resolution	Threshold	Precision	Recall	f1 score
160x128	0.50	**0.7608**	**0.7767**	**0.77**
	0.60	0.6953	0.6465	0.67

5.5.4 Threshold Behavioral Analysis

Threshold has been known as a border of transgression in between precision and recall values. On the general plane, the threshold does not impact the f1 score too greatly. The impact is on the borderline in between the recall and precision. In fact, the higher the thresh value used, the higher is the precision and recall becomes smaller in its stead, keeping the f1 score evened out. On the other hand, as the thresh decreases, the recall hits remarkable values all the while keeping precision at incredible low averages. At this point, this is known deep learning knowledge.

Three threshold values were tested roughly - 0.3, 0.50, 0.7. From the results obtained, the 0.50 seems to be the value that balances the best the relationship between precision and recall, as it was expected. Whether the desired outcome is to delineate the best area without having actual, pixelwise, high precision, a smaller threshold must then be used. If the desired output is the opposite, the contrary behaviour must be followed. Even if these follow through, the recall can achieve staggering values over 90% whilst the precision seems to be limited by an upper limit a little short of 80%. That would be the natural borderline of what the model can learn at most.

For analytic reasons, in all experiments, a pixel is considered labeled as 1 if the logit is above the 0.50 marked threshold. The opposite would happen for class 0. However, if the HGNet were to be used as a region proposal, allied with other methods, recall would need to be necessarily high. As such, on that specific case, it would be of interest to test a threshold of 0.3.

For the model optimization, in order to confirm the most optimal thresh to use, a directed experiment was made on threshold values. Table 5.1 showcases the difference in results between using a threshold of 0.60 or 0.50. Since the 0.5 demonstrated having the best balance in between precision and recall, alongside with better results, that value was used for all experiments explored on posterior sections.

For more detailed information on the threshold variation training sessions, refer to annex B.2.2.

5.5.5 Data Augmentation

The use of methods for data augmentation are applied in a different focus rather than the traditional in order to study their own possibilities of effectiveness.

The first approach for data augmentation is the insertion of random points inside a height grid in order to ensure the model is strong against noise in the pointcloud. This is done on the preprocessing of the input and points are created inside the boundaries of the grid. Since the coordinates are chosen at random, they may be completely outside of the frustum angled area.

Furthermore, another approach was taken for data augmentation. Since the rotation of the frustum was kept through the height grid, most training examples might end up being rotated over

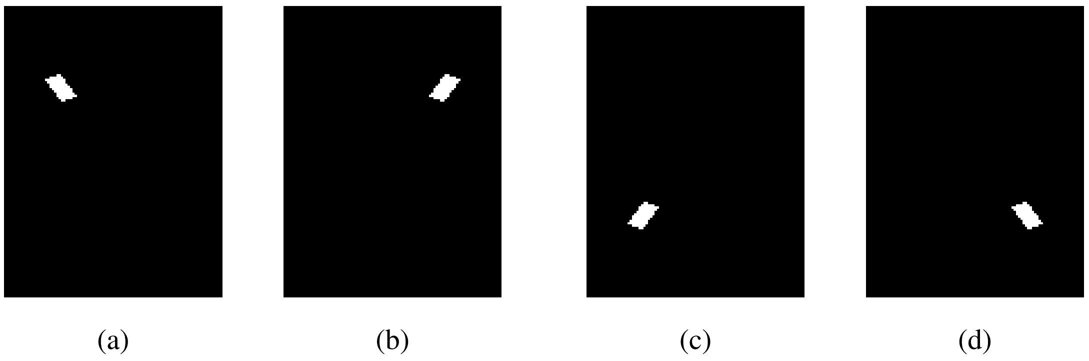


Figure 5.7: Reversion techniques for data augmentation. (a) Original labeled grid. (b) Horizontally inverted labeled grid. (c) Vertically inverted labeled grid. (d) Vertical and horizontally inverted labeled grid.

a certain angle. In order to make the network more robust towards this, a tunable percentage of the samples on each batch are repeated but reversed height grids which are created at training time.

Frustums go through depth which means that, in a height grid, they will have a disposition from the bottom to the top. Reversing the grid means that the depth variance is inverted and so the image will have a gradient from top to bottom. This helps the model be invariant to the disposition of the frustum on the plane. These reversions are made through the x and y axis. Graphically, these flips on the axis correspond to the inversion of the image horizontal and/or vertically. These inversions may be done only on one axis or on both simultaneously. Since the grids are hard to distinguish through the naked eye, some examples of reversed grids are presented through the filled box label style on figure 5.7.

5.5.6 Visualized Extractions

A system of saving predictions on some validation set examples over epochs of the training process was implemented in order to better understand what the model is specifically learning.

Through this process, the debug is made automatically on the labels, grids and learnt features. One example of the visualization of these improvements is shown on figure 5.8. For more examples, refer to annex B.1.

The first observation that was made was the fact that, on the height grid representation, it is almost impossible to discern through the naked eye where the points are to be found. The representation on png format decreases even more on the quality of shadow discernment. However, even on the array representation, the variations of value in between heights are quite small on most grids for it to have a significant impact on the grayscale gradient variation. Despite this difficulty, the network seems to be learning to have good sensibility to small float variations in height. Even on the hard examples, the model learns to identify a more prominent area on which an object might be found.

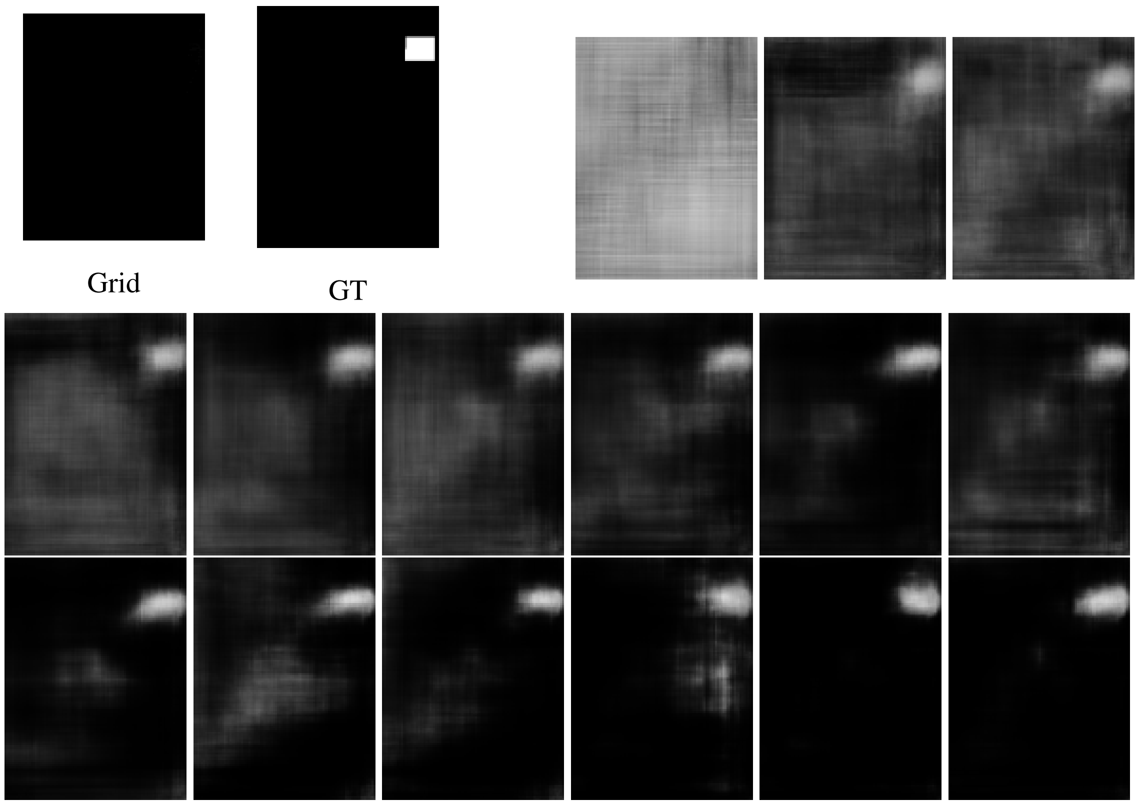


Figure 5.8: Model's prediction through epochs on an image from the validation set with the input grid and ground truth label showcased on the upper left corner.

Table 5.2: Recall and precision obtained through different grid dimensions with pixelwise object labels.

Label type	Resolution	Loss	Precision	Recall	Detection time (s/frustum)
Object pixelwise	96x64	1423.5092	0.4950	0.7222	0.0156
	128x96	1579.3886	0.4890	**0.7279**	**0.0104**
	160x128	1394.0702	0.4545	0.7051	0.0149
	192x160	830.2281	**0.5021**	0.6361	0.0184
	320x288	381.9638	0.5018	0.5005	0.0485

5.6 Experiments with HGNet

In order to properly analyze and incrementally optimize the model, some experiments were carried out with focus on different aspects of the network. The first area of action was to understand the impact that the size of the grid had on the training process and evaluation results. Moreover, it was found interesting to study the behaviour of the addition of data augmentation techniques.

Even more, the focal loss is also studied in terms of both stability, training outcomes and how it can impact the training of models for unbalanced classification tasks.

Through optimization procedures, it is found that batch normalization, even though advantageous for most models, impacts negatively the accuracy in terms of both recall and precision. Furthermore, learning rate is kept at an uncommon large value, 1e-4, used alongside with the Adam algorithm for gradient optimization. Both choices were made after heavy experimenting with the network's hyper parameters. These are kept through the variations made over the next sections.

5.6.1 Grid Dimensions

Table 5.2 and 5.3 showcase the results obtained for recall and precision with validation after training with different grid dimensions. Aside from this, the labeling styles discussed on previous sections are analyzed through out some label specific experimentation. On the ones identified as *object pixelwise* labeling, the labels used for metrics evaluation are the input image zero valued with only classification 1 at pixels occupied by an object's coordinate based point. On the other hand, when the methodology is identified as *filled box*, the labels are the entire oriented bounding box over a black background.

Through the analysis of table 5.2, loss value shows signs of better stabilization with the increase in dimensions. Even if precision seems to be non affected by the grid size, recall increases as the resolution of the grid gets smaller. The time necessary to process a frustum suffers with the increased grid dimensions generally, as expected. Even so, the time necessary for detection on each frustum is rather slim. This labeling format does not seem to be too effective for training and evaluation operations. Precision is not able to go beyond 0.5 due to the fact that the algorithm extracts areas better than exact pixelwise classification and comparison is ensued in between the bounding box area and object pixel.

Table 5.3: Recall and precision through different dimensions with filled box labeling style.

Label type	Resolution	Loss	Precision	Recall	Detection time (s/frustum)
Filled box	128x96	4213.1544	0.5776	0.6016	**0.0123**
	160x128	4511.3496	**0.7279**	**0.6966**	0.0213
	320x288	4135.5612	0.5432	0.5974	0.0407

It is reasonable to conclude that precision values are much higher when using filled labels. This has a simple explanation. As it has been discussed, due to the fact that the algorithm outlines areas rather than pixelwise driven segmentation leads to a smaller precision when evaluating over only bordered pixel bounded segmentation. On the other hand, bounding boxes represent an area of interest that can be successfully and more precisely removed by the base model. So, using filled labels seems to be more advantageous for backbone training with this model. Besides this, the areas extracted also feature an orientation which might be useful for final bounding box extraction. Table 5.3 showcases the results obtained on the filled box experiments. First of all, the prediction of filled labels takes a larger portion of time over the prediction of the border even if slightly. Validation loss values are quite large. However, during training, the focal loss is much more stable with the filled box labeling style.

From table 5.3 and from the bundle of resolution experiments performed, the $160x128$ resolution was found to be the most effective to bring out good result pools. This may be explained by the architecture of the network. As it can be denoted through figure 5.4, the contraction path diminishes the image until it reaches a 1/16 of its initial size. For the dimension $160x128$, this produces perfectly sized results since it downsizes to the $10x8$ size. Through these results and due to the fact that the smaller the image size the faster would be training and segmentation procedures, it was chosen to keep the $160x128$ resolution on follow up experimentation.

For detailed plots of the training procedures through different resolutions please refer to annex B.2.1.

5.6.2 Focal Loss Analysis

Focal loss involves two hyper parameters that can be adjusted, the alpha and gamma. So, as to better comprehend how these might affect training and evaluation operations, some experiments are performed with different alpha/gamma values.

Through training, it has been noticed that sometimes the focal loss is not too stable on the binary classification ensemble when using certain parameters. For example, as it can be denoted on table 5.4, when the α value is higher than 0.9, the training goes astray, resulting on the tendency showcased by the model to select all elements as the non active class. For exhaustive plots of the training and validation losses, refer to annex B.2.3.

At the same time, it can be observed through table 5.4 that the alpha balancing factor has an impact on the relationship between precision and recall. In fact, as values approximate zero, the

Table 5.4: Results of precision and recall with the variation of the α parameter.

γ	α	Precision	Recall	f1 score
	2	0.01345	**1.0**	0.027
	1	0.01497	**1.0**	0.029
	0.9	0.6306	0.8657	0.73
2.0	0.75	0.7608	0.7767	**0.77**
	0.6	0.7518	0.7688	0.76
	0.4	0.8174	0.6899	0.75
	0.25	**0.8501**	0.6520	0.74

tendency is to increase the precision whilst the recall suffers a decrease. This relationship is best visualized through graphic 5.9.

With the increasing α value, precision tends to drop whilst recall increases. The point at which both values almost meet is the ideal parameter value since it holds the common maximum for both recall and precision. This ideal would be around 0.75. Another aspect of the relationship that can be denoted through the plot is that even if the recall only varies inside a small interval (approximately from 0.65 to 1.0), the precision can be much more greatly affected by the parameters chosen for the focal loss. In fact, precision varies throughout almost the entire possible scale, only featuring an upper bound situated at 0.85. This showcases the capacity that the model is great at choosing the relevant pixels.

Building up from this experiment, the α value was kept at 0.75 and some variations on to γ were tested. Table 5.5 showcases the results obtained. Using a value of 0.5 for γ makes training go astray after a bundle of epochs. The loss suddenly hits nan values and no more learning is possible. However, values above 2.0 showcase good results with even f1 scores. However, training is more unstable with values larger than 2.0. For this reason, the adopted γ factor is of 2.0.

Unlike α, the γ parameter does not seem to affect the training and validation procedures as much. In fact, recall and precision do not showcase such a dependence towards this loss parameter.

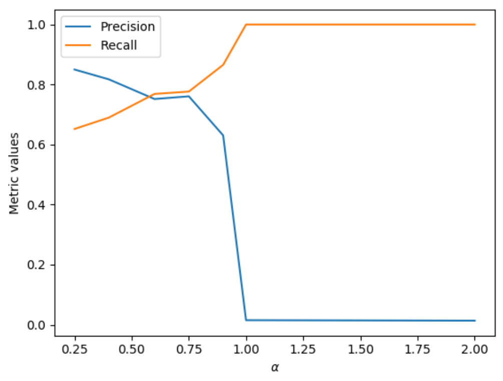


Figure 5.9: Precision and recall plots through different α values.

Table 5.5: Results obtained in precision and recall through variations on the γ factor.

α	γ	Precision	Recall	f1 score
	0.5	0.6510	0.6761	0.66
0.75	2	**0.7608**	0.7767	**0.77**
	5	0.7542	**0.7808**	**0.77**

This would be expected since both parameters have different objectives. The γ parameter is more oriented towards the balancing of under sampled classes and as such has the greatest impact on the loss values obtained and the smoothness of the training procedures.

For further insight into the training procedures for the focal loss γ variation, refer to the annex B.2.4.

5.6.3 Data Augmentation Factor Analysis

It is of interest to grasp how effectively the data augmentation methodologies can have an impact on final accuracy. For this purpose, a data augmentation factor is added to the network hyper parameters. This factor is balanced in between 0 and 1. Its value represents the percentage through which to augment the original training data. On each batch, the tunable percentage will be some samples at random flipped at random as well. This means that any sample might be picked and any reversion might be applied, either only vertical or horizontal or both simultaneously.

The effects of data augmentation are then studied through the realization of training sessions with different augmentation factors. The results can be observed through table 5.6.

From the observation of the results, even if the precision wavers around similar values, the recall does get a boost due to the presence of more data. Eventually, this leads the model to achieve better performance generally. On this case, it can be seen that it only provides an increment of around 0.1 0.2 points in score. The relationship between the factor used for data augmentation and the increase of the results is not straightforward. In fact, as it can be observed from table 5.6, the use of a factor of 0.2 brings about similar results as a factor of 0.5. However, more experiments are necessary in order to provide an informed conclusion.

For a more in depth view of the training processes for the data augmentation factor, refer to annex B.2.5.

Table 5.6: Results obtained with different augmentation factors.

Aug factor	Precision	Recall	f1 score
0.0	0.7608	0.7767	0.77
0.2	**0.7499**	**0.8112**	**0.78**
0.5	0.7429	0.7967	0.77

5.7 Segmentation Speed Analysis

Since detection times are absolutely invaluable for application on the scenario of autonomous driving, it is of interest to analyze the timing values by frustum.

Figure 5.10 showcases the exponential relationship in between the summed segmentation time and the number of frustum over several different grid dimensions. The summed segmentation time is the time to segment all resulting frustum present on a sample, considering that a sample is from the full pipeline and features an unknown number of objects. All times used for calculations on segmentation time include the creation of the height grid, model compilation, prediction and evaluation. The hardware used to extract the statistics was the already before mentioned NVIDIA GeForce GTX 1080 Ti.

Since all segmentation experiments were carried out on the object dataset, the statistics from that specific dataset are considered. If we recall the figure 4.2, from the previous section, a sample from that specific bundle of data would never have over 20 objects on the same frame. From the segmentation times extracted, this would mean that, at a standard grid resolution of $160x128$ the summed segmentation time would be around 0.29 seconds if all frustum are processed separately.

Basing the assumption once more on figure 4.2, most frames include 1 to 6 objects. On that case, with a resolution of $160x128$, the summed time would be between 0.01489 and 0.08934 seconds. The segmentation times would only feature such high values in the case that every single frustum is processed one at a time.

Timing constraints are not the objective of this work. However, from the top of the head, estimating that most GPU's are able to parallel tasks in the order of 10's of thousands, this would mean that the time needed to segment a frame, even with 20 objects, would be approximately equal to that of segmenting a sample with only one frustum removed.

Observing now other aspects of figure 5.10, it is demonstrated clearly that the higher is the grid dimension, the higher will be its segmentation time. This is naturally expected since the more resolution is added to the grid, the more computational resources would be necessary to create and process it.

5.8 Summary

This chapter showcases the progress made whilst researching new methodologies for exploration of both representations and segmentation of LiDAR pointclouds. Through experimentation of machine learning such as the $k-$means algorithm and eigenvector analysis, some conclusions are drawn.

An occupancy grid representation is proposed, inspired by the modulated RGB channel elevation grid proposed by Yu *et al.* [YWH+17]. This grid samples pointcloud points through the bird's eye view perspective and compiles them on a flat, grayscale image which showcases the maximum elevation found at any cell. A methodology for search space reduction is also adopted, based off of Qi *et al.* [QLW+17], through the removal of pointcloud frustum.

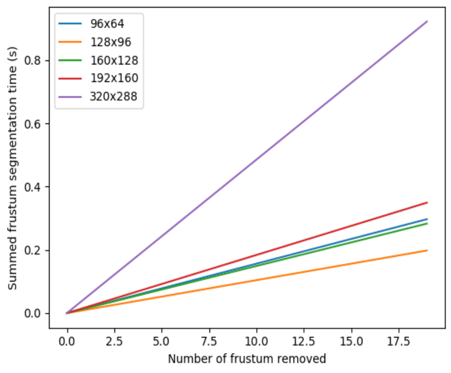


Figure 5.10: Segmentation time per number of frustum over several grid dimensions.

Furthermore, a novel network is explored for segmentation on the height grid representation. This network aims at classifying pixels as active or non active accordingly to the pixel belonging to the object's area or not in a frustum. Aside from presenting the architecture, several experiments are carried out on different configurations and settings in order to ascertain the network's capabilities. The focal loss was used for the training procedures. Its parameter variation is also delineated through experiences on the HGNet.

Chapter 6

Conclusions and Future Work

Through the course of this project, many experiments were carried out and varied, intricate conclusions were drawn after discussion and observation. This section serves the purpose of enlightening the reader on a general plane over the achievements made by this work, what came to be understood and what will come to be explored. As such, the satisfaction of the objectives is first delivered in the form of a summary on the work that was made during the project's development time. A critical analysis is then carried out in order to point out what was successful and what would need further development. Besides that, the future work is dwelt on, in order to give a grasp of what are the topics of interest that were found to be promising moving forward.

6.1 Main Contributions

The first and most prominent objective of the work at hand was to explore methodologies for object detection in autonomous driving scenarios. Throughout the course of the development phase, dozens of models were tested and analyzed on the context of unmanned driving processes. Even going further from the initially imposed objectives, this work extended the scope from only working with the LiDAR sensor to including RGB camera images as well.

At first, several existent models, whether generalized or specifically applied were explored on the specified setting in order to ascertain their own effectiveness and adaptation possibilities towards these issues. On this topic, the YOLO image detector was tested with the KITTI dataset, being able to detect up to eight classes effectively on basic, standard scenarios. Many fold configurations were tested on this detector which eventually lead to the joining of the ideal conditions and a guide for configuration of the model on the general plane. Furthermore, it was even proved how YOLO could generalize to other sensory data without much training and/or further configuration hassle such as grayscale imagery. The submission to the benchmark, however, did not showcase results on par with what would be expected of the model, bringing the conclusion that it is unfit for many recurrently incurring road situations.

Moving towards 3D sensory data, machine learning techniques were explored for a wide range of tasks. Starting from pointcloud processing and analysis methodologies, contributions were made on reduction of object's search space, eigenvector based analysis and ground segmentation. A bundle of possibilities were explored and a guide is presented over the directions to follow and what not to test since it brings out unsatisfying results.

More importantly, a new representation for LiDAR is presented and analyzed in terms of qualitative potential for posterior bounding box regression. The designed height grid can be though of as a relief map in bird's eye view. Sources of errors for this representation are described alongside with the great advantages to be gained with its use.

Moreover, a new model is proposed for object area segmentation working on the height grid representation. This model achieves a staggering f1 score of 0.78 for images with 3 times smaller resolution than the standard image recognition algorithms dare to work with. This leads to a fast training process which means the network can be easily extrapolated to any kind of scenario with little processing overhead. The proposed HGNet is not only able to achieve good pixelwise segmentation. In fact, it is able to delineate the bounding box area with orientation information from a small sample of points taken from the LiDAR in bird's eye view. At the same time, the HGNet showcased a great segmentation speed by frustum which contributes to the possibility of using the model on real time scenarios. From the outputs of the network, bounding boxes can be easily derived with further simple assisting processes.

On the overall plane, a possible pipeline for 3D object detection is proposed which mixes information from two types of sensors. This pipeline leverages a simple, light and fast fusion of the 2D information extracted through both cameras and LiDAR bird's eye view in order to bring real 3D powered bounding boxes. Overcoming some shortcomings of similar pipelines, the process for the use of joint information is idealized in a way that makes the most use of both sensors.

To sum up, contributions were made both on the field of 2D image based object detection and LiDAR processing. The objectified view of exploring new uncharted areas or methodologies was also fulfilled, having research been carried out with no fear of testing out of the box concepts. As such, all objectives are deemed as concluded and with great achievements through them.

6.2 Final Remarks

Even through all that was achieved, one can still distinguish between the strong and weak points on the work that was done. A critical analysis is always necessary in retrospective so that one has an idea of what has been done right and what needs further development.

As such, the first denoted weakness was the reduced testing of camera detection models. Many more models exist and particularly, the YOLO model has been known to be falling behind compared with other general purpose detectors. Further than the general plane, there are several models that are specifically built for the autonomous driving scenario. Having a comparison analysis with these models would be of interest for the designed objectives.

On another note, the proposed pipeline can certainly achieve the desired outcomes. It can successfully leverage the use of both sensors with a concise, simple flow. However, the fact that the procedures are carried out as individual steps contributes to the complexity of training all models separately and adds to the detection time since each frame needs to go through processing steps in between models. Furthermore, some of the parts of the pipeline are not possible to optimize as it is. For example, parallel processing is impossible before the frustum is removed. Besides that, objects can only be found on LiDAR if they are found on camera beforehand. This eventually means that the final pipeline will have a mean average precision with an upper bound defined by the 2D model's own mAP.

On another topic, the height grid explored for LiDAR segmentation creates an obstacle on certain rare cases. When the object is close to a corner of the image, it may happen that a bounding box is rotated towards the outside of the image field of view. On this case, it would be harder to regress oriented bounding boxes from the grid image only even if the points belonging to the object are all inside the image. On this precise situation, the HGNet would then loose some quality of the outputted information since the segmentation would not be helpful in the guessing of the orientation and exact boxed coordinates.

The proposed HGNet has brought extensively impressive results. In fact, it is able to extract meaningful features from an image with such small variations in height. Since the frustum are normalized, the height values sometimes end up having variations only starting on the fourth numeric float point. This would be another weakness of the height grid. Furthermore, since the height grids are not formalized in between each other, the model needs to learn through completely different settings each time. However, even with such limitations, the model was able to reach an F1 score of 0.78 which is incredibly high considering it extracts areas and not pixelwise activation. Moreover, depending on the objective at hand, it has been shown that this network can be tuned to have outstanding recall over some precision cost or vice versa. The fact that the frustum is imprinted on the images with its original rotation does not help make the model invariant to rotation of the data. This would be another weakness provoked by the pre processing steps. Furthermore, ideally, more optimization experimentation would have to be done on the HGNet in order to truly have it achieve its full potential. For example, the effects of data augmentation are not fully documented yet and the study of gradient optimization algorithms could only benefit the end results.

6.3 Future Work

Building up from what was considered to be lacking, the following paths of improvement would be of interest to follow. For example, the poor performance of YOLO on the benchmark brings forward the question on whether the general purpose object detectors can be adapted to such a demanding environment as the autonomous driving one. On that thought, it would be of interest to explore the RetinaNet, currently the general purpose state-of-the-art model, so as to closely assess whether the weakness showcased is directly related with the YOLO model or with the general approach taken by such algorithms.

On the topic of the pipeline, it is identified the optimization of the whole flow through the creation of a model capable of leveraging both camera and LiDAR information at the same time. One great improvement could certainly be achieved whilst researching language transgression methodologies in between camera and LiDAR, of the style of the joint model proposed by Redmon *et al.* [RF17]. If a model could join the models through one unique processing flow, it would be of interest to find a common transgression in between both spaces.

Furthermore, the height grid could be improved thought the transformation on RGB images as it is proposed on [YWH$^+$17]. The exploration of several methods for channel use would be of interest aside from features such as elevation. Further than that, it is also identified a possibility of evolving the HGNet through a more FPN [LDG$^+$17] styled approach. It is idealized that several grids could be leveraged with channels dedicated to different features such as height, density and intensity for example. Using then those grids for an entranced, collaborative feature extraction would be a potential filled direction of study. On that specific note, it would also be of interest to test another architecture through the addition of short connections in between the mid steps of the contraction and expansion path, in the fashion of the U-Net [RFB15].

On a more general setting, this work would allow the further study of the joint pipeline. It is desirable that the extra steps left on the pipeline would be implemented on a future perspective. This would include adding complexity to the HGNet with the use of the logit segmentation in order to regress bounding boxes with orientation.

Other identified projects moving forward would be the start of object classification. As it has been mentioned beforehand, it would be interesting to develop a mixed camera and LiDAR object identification model that could classify a bundle of categories from both sensory information. Even further than that, moving on to the tracking scenario, a possible direction of exploration would be the use of both sensory data in order to keep track of all objects even when it would leave a certain sensor's field of view.

To sum up, many contributions were made to the field. With the exploration of existent and new methods for object detection in both camera and LiDAR, this work has dealt with the topic of autonomous driving as the huge research question mark it is. Moving forward, more experiments and analysis can be made on the explored topics with the perspective of improving their shortcomings as well as completely new paths that seem to have the potential to bring out great results.

References

[04] The princeton shape benchmark. In *Proceedings of the Shape Modeling International 2004*, SMI '04, pages 167–178, Washington, DC, USA, 2004. IEEE Computer Society.

[AGP+17] A. Asvadi, L. Garrote, C. Premebida, P. Peixoto, and U. J. Nunes. Depthcn: Vehicle detection using 3d-lidar and convnet. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, Oct 2017.

[Ale18] AlexeyAB. Yolo-v3 and yolo-v2 for windows and linux. `https://github.com/AlexeyAB/darknet`, 2018. [Online; accessed 13-June-2018].

[APPN16] Alireza Asvadi, Cristiano Premebida, Paulo Peixoto, and Urbano Nunes. 3d lidar-based static and moving obstacle detection in driving environments. *Robot. Auton. Syst.*, 83(C):299–311, September 2016.

[BNB17] A. Börcs, B. Nagy, and C. Benedek. Instant object detection in lidar point clouds. *IEEE Geoscience and Remote Sensing Letters*, 14(7):992–996, July 2017.

[BP16] Axel Bender and El'ias Marel Porsteinsson. Object Classification using 3D Convolutional Neural Networks. *Chalmers University of Technology*, 2016.

[BSC12] Jens Behley, Volker Steinhage, and Armin B Cremers. Performance of Histogram Descriptors for the Classification of 3D Laser Range Data in Urban Environments. *IEEE International Conference on Robotics and Automation*, pages 4391–4398, 2012.

[CCR+17] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1827–1836, July 2017.

[CDB18] Nesrine Chehata, Nicolas David, and Frédéric Bretar. Lidar data classification using hierarchical k-means clustering. 06 2018.

[CZMK16] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. 02 2016.

[EEV+14] Mark Everingham, S. M Ali Eslami, Luc Van Gool, Christopher K I Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 2014.

[FB81] Martin a Fischler and Robert C Bolles. Paradigm for Model. *Communications of the ACM*, 24(6):381–395, 1981.

[FKG13] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[FLR$^+$17] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C. Berg. Dssd : Deconvolutional single shot detector. *CoRR*, abs/1701.06659, 2017.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[Gir15] Ross Girshick. Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-October:2980–2988, 2017.

[HL09] M. Himmelsbach and T. Luettel. Real-time object classification in 3D point clouds using point feature histograms. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 994–1000, 2009.

[HSL$^+$17] Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan Dirk Wegner, Konrad Schindler, and Marc Pollefeys. Semantic3d.net: A new large-scale point cloud classification benchmark. *CoRR*, abs/1704.03847, 2017.

[JS16] Jing Huang and Suya You. Point cloud labeling using 3D Convolutional Neural Network. *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2670–2675, 2016.

[KM11] R. Kalam and K. Manikandan. Enhancing k-means algorithm for image segmentation. In *2011 International Conference on Process Automation, Control and Computing*, pages 1–4, July 2011.

[KML$^+$17] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Lake Waslander. Joint 3d proposal generation and object detection from view aggregation. *CoRR*, abs/1712.02294, 2017.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[KWB08] K. Klasing, D. Wollherr, and M. Buss. A clustering method for efficient segmentation of 3d laser data. In *2008 IEEE International Conference on Robotics and Automation*, pages 4043–4048, May 2008.

[LDG+17] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, July 2017.

[LGG+17] Tsung Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal Loss for Dense Object Detection. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:2999–3007, 2017.

[Li16] Bo Li. 3D Fully Convolutional Network for Vehicle Detection in Point Cloud. 2016.

[LMB+14] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014.

[LSZ+16] Min Liu, Yifei Shi, Lintao Zheng, Yueshan Xiong, and Kai Xu. Volumetric spatial transformer network for object recognition. *SIGGRAPH ASIA 2016 Posters on - SA '16*, (1):1–2, 2016.

[LUG16] Stefan Lange, Fritz Ulbrich, and Daniel Goehring. Online vehicle detection using deep neural networks and lidar based preselected image patches. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2016-Augus(Iv):954–959, 2016.

[LV15] K. Lenc and A. Vedaldi. R-cnn minus r. In *British Machine Vision Conference*, 2015.

[LW13] Marco Langerwisch and Bernardo Wagner. Building variable resolution occupancy maps assuming unknown but bounded sensor errors. In *IEEE International Conference on Intelligent Robots and Systems*, 2013.

[LZX16] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. *CoRR*, abs/1608.07916, 2016.

[Mel15] T J Melanson. Using Faster-RCNN to Improve Shape Detection in LIDAR. *Stanford University*, 2015.

[MET17] D. Matti, H. K. Ekenel, and J. P. Thiran. Combining lidar space clustering and convolutional neural networks for pedestrian detection. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, Aug 2017.

[MGG17] Valentin Magnier, Dominique Gruyer, and Jerome Godelle. Automotive LIDAR objects detection and classification algorithm using the belief theory. *IEEE Intelligent Vehicles Symposium, Proceedings*, (Iv):746–751, 2017.

[MNG17] M. P. Muresan, S. Nedevschi, and I. Giosan. Real-time object detection using a sparse 4-layer lidar. In *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 317–322, Sept 2017.

[MS15] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, Sept 2015.

[MYLX11] Z Man, W Ye, P Lou, and H Xiao. Feature extraction based on split-merge in range image of lidar. 22:2303–2306, 10 2011.

[OK17] Sang-Il Oh and Hang-Bong Kang. Object Detection and Classification by Decision-Level Fusion for Intelligent Vehicle Systems. *Sensors*, 17(1):207, 2017.

[PJ17] Marek Pavelka and Václav Jirovský. Lidar based object detection near vehicle. *2017 Smart Cities Symposium Prague, SCSP 2017 - IEEE Proceedings*, 2017.

[Pro09] D.V. Prokhorov. Object recognition in 3D lidar data with recurrent neural network. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 9–15, 2009.

[QCS$^+$16] B. Qin, Z. J. Chong, S. H. Soh, T. Bandyopadhyay, M. H. Ang, E. Frazzoli, and D. Rus. A spatial-temporal approach for moving object recognition with 2D LIDAR. *Springer Tracts in Advanced Robotics*, 109:807–820, 2016.

[QLW$^+$17] Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. *CoRR*, abs/1711.08488, 2017.

[RCL$^+$17] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y. W. Tai, and L. Xu. Accurate single stage detector using recurrent rolling convolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 752–760, July 2017.

[RDGF16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, June 2016.

[RDS$^+$15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[Red16] Joseph Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016. [Online; accessed 05-May-2018].

[RF17] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, July 2017.

[RFA18] Joseph Redmon, Ali Farhadi, and Coco Ap. YOLOv3 : An Incremental Improvement. *Tech Report*, 2018.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[RHGS17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[RLR13] D. O. Rubio, A. Lenskiy, and J. H. Ryu. Connected components for a fast and robust 2d lidar data segmentation. In *2013 7th Asia Modelling Symposium*, pages 160–165, July 2013.

[RLW+09] Eric Richter, Philipp Lindner, Gerd Wanielik, Kiyokazu Takagi, and Akira Isogai. Advanced occupancy grid techniques for lidar based object detection and tracking. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 450–454, 2009.

[RQSMJG17] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[SSJ06] M. Szarvas, U. Sakai, and Jun Ogata. Real-time Pedestrian Detection Using LIDAR and Convolutional Neural Networks. *2006 IEEE Intelligent Vehicles Symposium*, pages 213–218, 2006.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[TRC+] TA Teo, JY Rau, LC Chen, JK Liu, and WC Hsu. a Spilt-and-Merge Technique for Building Shaping. *Loc.Geomatics.Ncku.Edu.Tw*.

[UVGS13] J R R Uijlings, K E A Van De Sande, T Gevers, and A W M Smeulders. Selective Search for Object Recognition. *Int J Comput Vis*, 2013.

[Wik18] Wikipedia. F1 score — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=F1%20score&oldid=841695054, 2018. [Online; accessed 13-June-2018].

[Wor12] Digital Camera World. Cheat sheet: how your digital camera turns captured light into an image, August 2012.

[WSD07] Thorsten Weiss, Bruno Schiele, and Klaus Dietmayer. Robust Driving Path Detection in Urban and Highway Scenarios Using a Laser Scanner and Online Occupancy Grids. *2007 IEEE Intelligent Vehicles Symposium*, pages 184–189, 2007.

[WSK+15] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.

[WT04] Miao Wang and YH Tseng. Lidar data segmentation and classification based on octree structure. *Parameters*, 2(1):1–6, 2004.

[XMS14] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision*, pages 75–82, March 2014.

[YWH+17] S. L. Yu, T. Westfechtel, R. Hamada, K. Ohno, and S. Tadokoro. Vehicle detection and localization on bird's eye view elevation images using convolutional neural network. In *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pages 102–109, Oct 2017.

[ZATX03] Sen Zhang Sen Zhang, M. Adams, Fan Tang Fan Tang, and Lihua Xie Lihua Xie. Geometrical Feature Extraction Using 2D Range Scanner. *2003 4th International Conference on Control and Automation Proceedings*, (June):10–12, 2003.

[ZF14] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing.

[ZT17] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CoRR*, abs/1711.06396, 2017.

Appendix A

YOLO Architecture and Processing Breakdown

An in depth view into the workings of the YOLO model.

A.1 Version 1 Full Architecture

Full architecture of the first version of YOLO presented on Redmon et al. first's paper[RDGF16] is showcased on figure A.1.

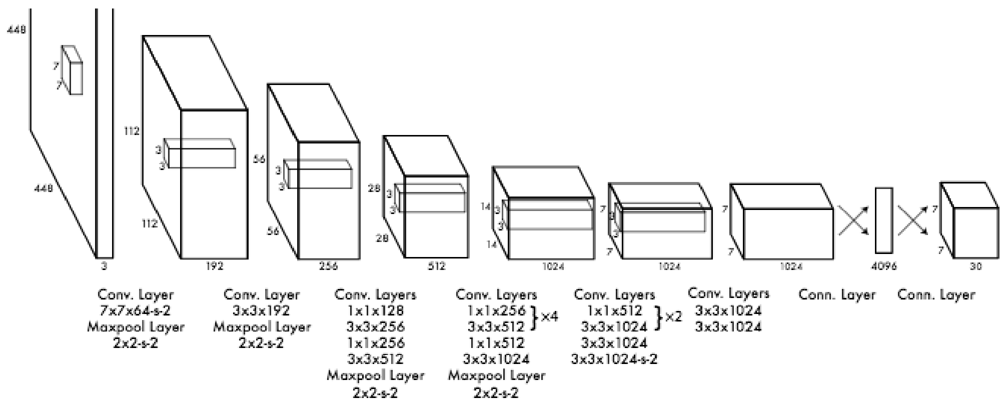
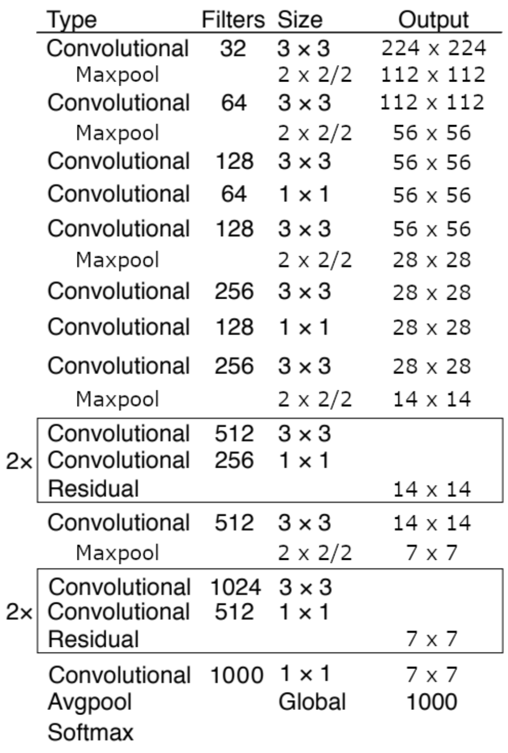


Figure A.1: YOLO version 1 full architecture breakdown. Adapted from [RDGF16].

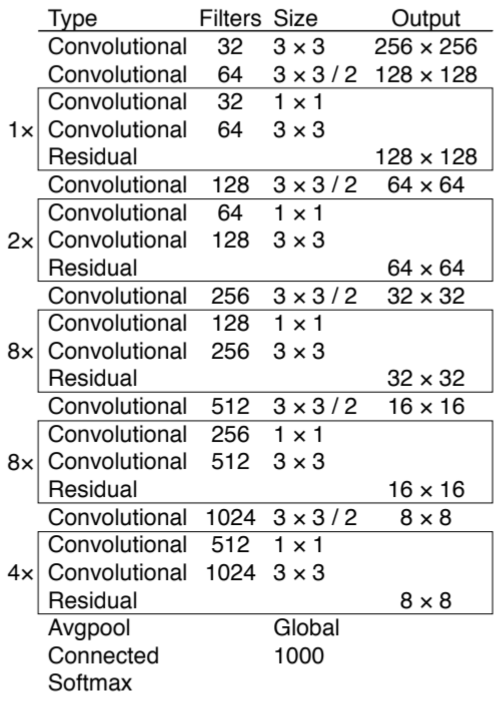
A.2 Darknet-19 and Darknet-53

Figure A.4 shows the evolution in architecture of the classifier named Darknet. Going from 19 to 53 layers, the newest Darknet[Red16] version is much more complex.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	224 x 224
	Maxpool		2 x 2/2	112 x 112
	Convolutional	64	3 × 3	112 x 112
	Maxpool		2 x 2/2	56 x 56
	Convolutional	128	3 × 3	56 x 56
	Convolutional	64	1 × 1	56 x 56
	Convolutional	128	3 × 3	56 x 56
	Maxpool		2 x 2/2	28 x 28
	Convolutional	256	3 × 3	28 x 28
	Convolutional	128	1 × 1	28 x 28
	Convolutional	256	3 × 3	28 x 28
	Maxpool		2 x 2/2	14 x 14
2×	Convolutional	512	3 × 3	
	Convolutional	256	1 × 1	
	Residual			14 x 14
	Convolutional	512	3 × 3	14 x 14
	Maxpool		2 x 2/2	7 x 7
2×	Convolutional	1024	3 × 3	
	Convolutional	512	1 × 1	
	Residual			7 x 7
	Convolutional	1000	1 × 1	7 x 7
	Avgpool		Global	1000
	Softmax			

Figure A.2: Darknet-19 architecture.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure A.3: Darknet-53 architecture.

Figure A.4: Darknet-19 and Darknet-53 architectures. Adapted from [RF17] and [RFA18] respectively.

A.3 Training Evolution

Figure A.5 features the YOLO loss evolution over iterations from the training session with the object dataset.

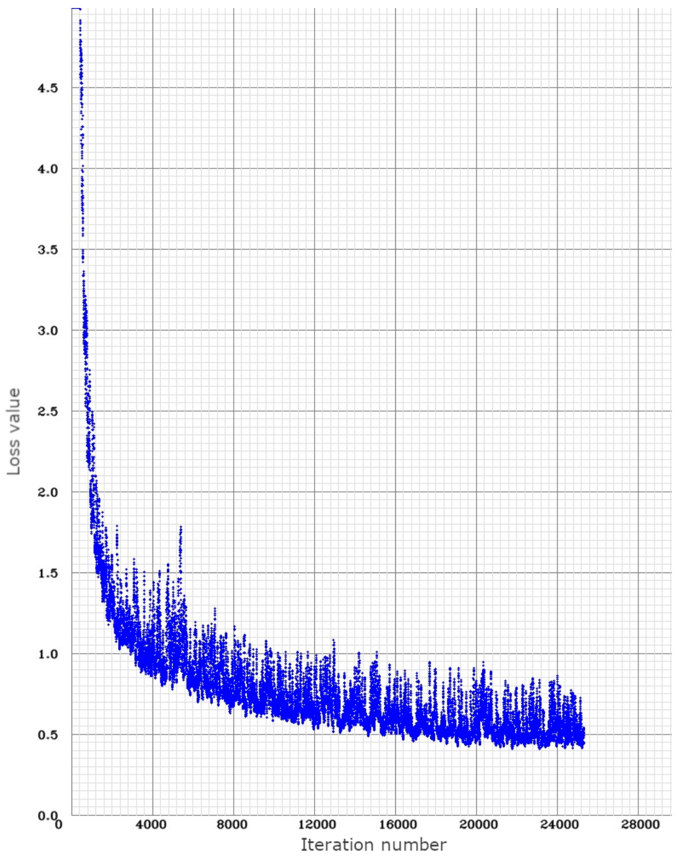


Figure A.5: YOLO training loss evolution chart.

YOLO Architecture and Processing Breakdown

Appendix B

HGNet Training Observations

A more in depth view of the learning process through epochs of the HGNet are presented through some exhibits.

B.1 Evolution Over Epochs

Predictions of the model by epoch when evaluating on the test set. Figures B.1 and 5.8 showcase two different grids from the test set. The grid image is imperceptible on the png imaging format.

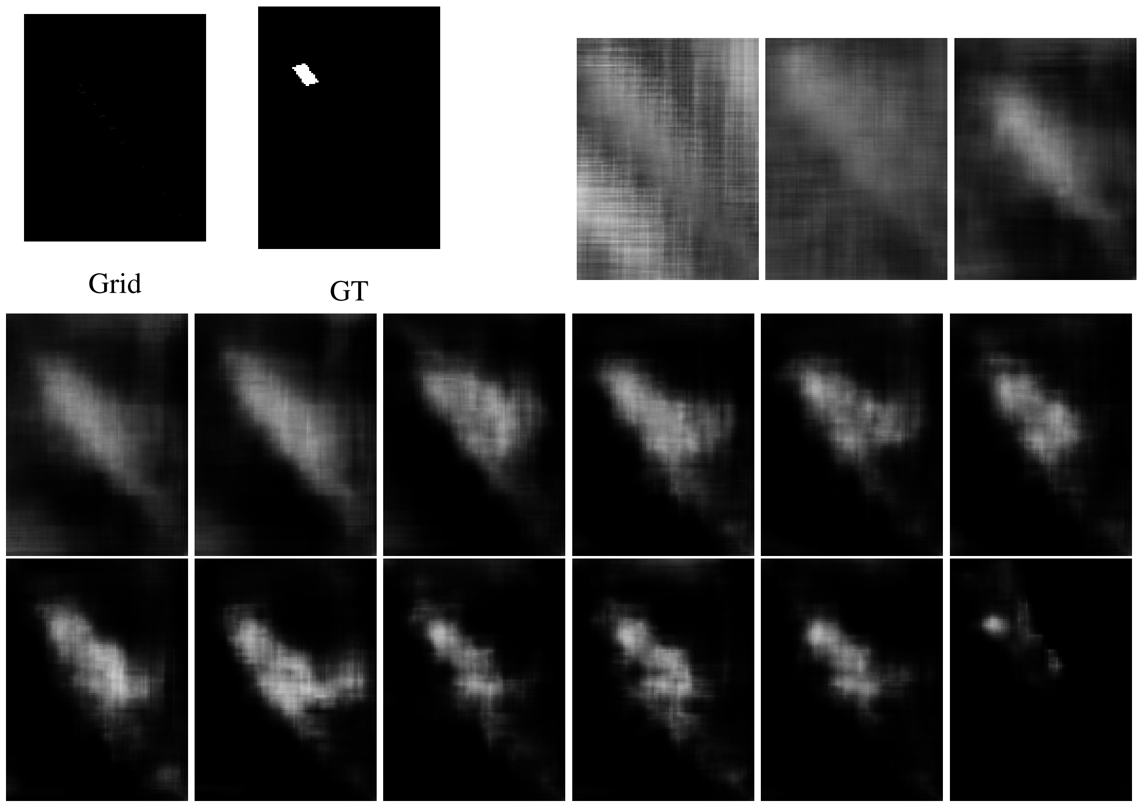


Figure B.1: Model's prediction through epochs on an image from the validation set with the input grid and ground truth label showcased on the upper left corner.

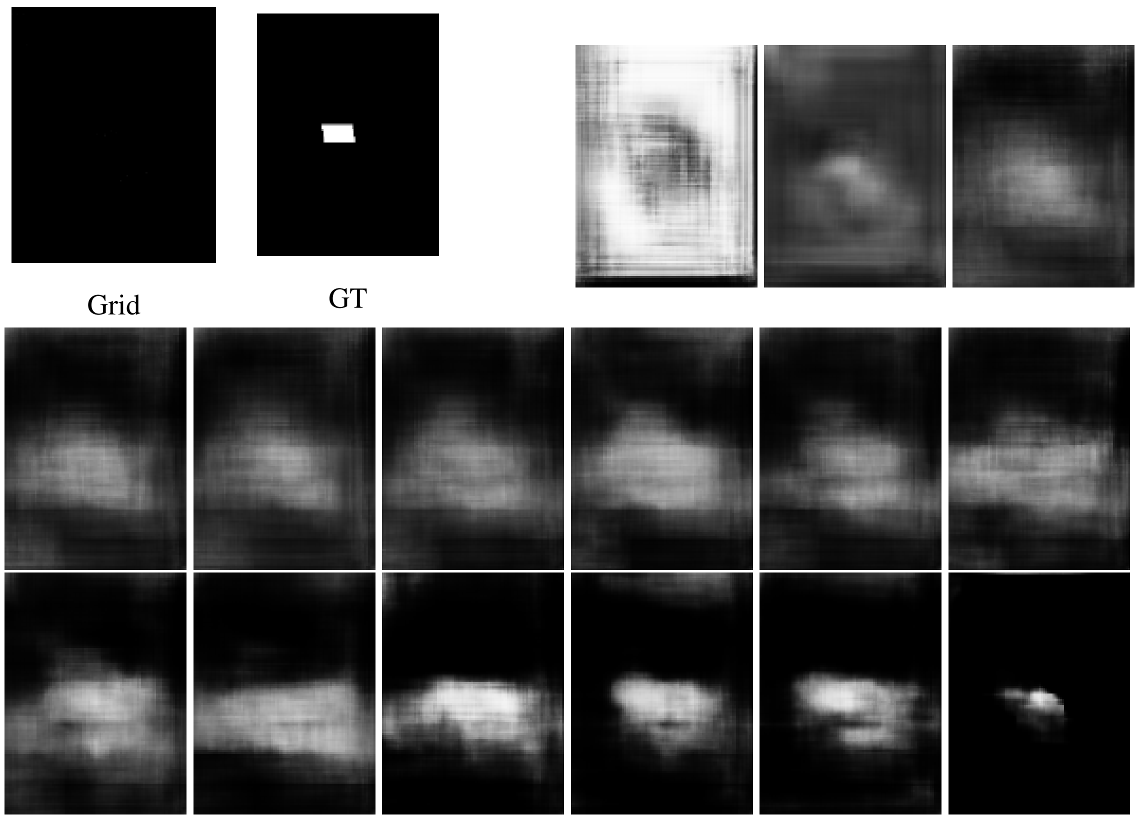


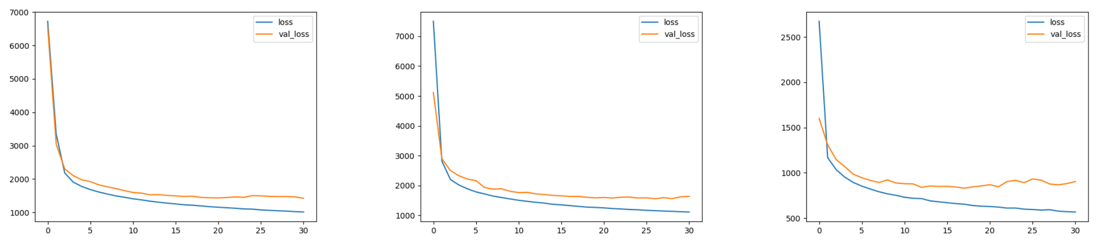
Figure B.2: Model's prediction through epochs on another image from the validation set with the input grid and ground truth label showcased on the upper left corner.

B.2 Training Processes

Compilation of the training/validation loss plots obtained through different experiments. Plots were drawn each 10 epochs and the maximum shown epoch does not correspond to the point at which values were taken for evaluation purposes. A early stopping time is identified for all the training sessions.

B.2.1 Grid Dimensions Tuning

Graphics showcased on B.3 demonstrate the smooth training processes throughout different dimensions. The loss drops constantly until it stabilizes due to the use of the focal loss. Even though the increased dimensions of the grid stabilize the loss to smaller values, the evaluation loss gets a little more rocky.

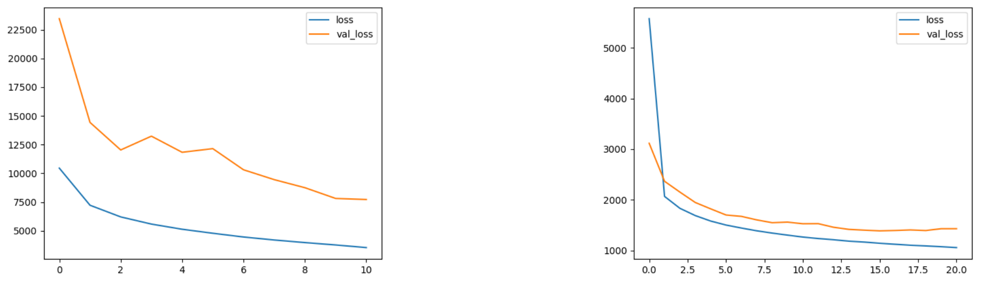


96x64 128x96 192x160

Figure B.3: Training loss and evaluation loss plots for several resolutions of the HGNet.

B.2.2 Threshold Tuning

From the two threshold values specifically tested, the plots showcased on figure B.4 of the training and validation losses were obtained.

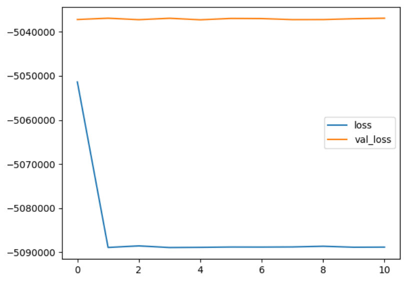


thresh = 0.6 thresh = 0.5

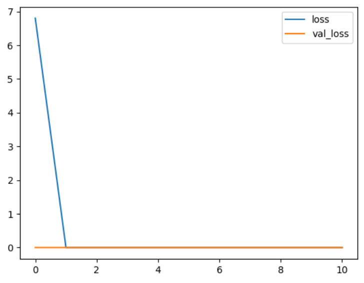
Figure B.4: Training loss and evaluation loss plots from the HGNet threshold tuning experiments.

B.2.3 α Tuning

Using α values over 0.9 spoils training. With a value of 1, training and evaluation losses reach zero and stay there through all epochs. With a larger value, of 2 on this specific case, the loss is negative for both training and evaluation procedures, with no effective change after the minimal values have been attained. Figure B.5 showcases these relationships.



$\alpha = 2$

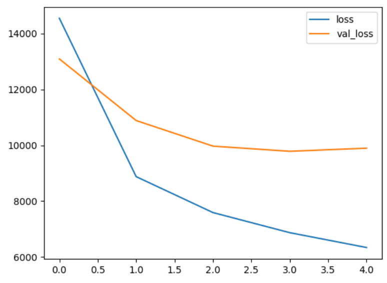


$\alpha = 1$

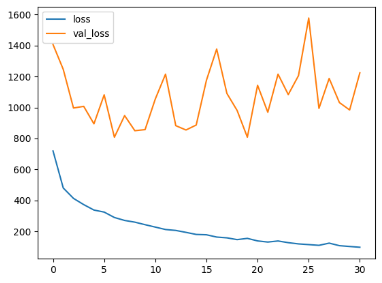
Figure B.5: Training loss and evaluation loss plots over different α values.

B.2.4 γ Tuning

Making use of γ values under 2 seems to lead training astray as it is represented by figure B.6 through the existence of loss only until epoch 4 when it suddenly turns nan. Values over 2.0 are unstable provoking several peaks on loss value through training. However, in general, the loss values obtained are much smaller, being at the border of the hundreds.



$\gamma = 0.5$

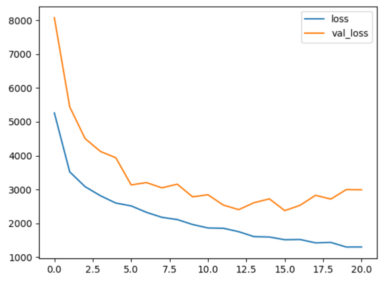


$\gamma = 5$

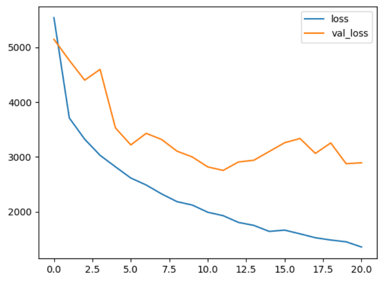
Figure B.6: Plots of the training and validation loss originated with different γ parameters on the loss function.

B.2.5 Augmentation Factor Tuning

As it can be denoted from figure B.7, the training is more smooth when using an augmentation factor of 0.2. Loss wavers around quite more with a doubling data factor.



$factor = 0.2$



$factor = 0.5$

Figure B.7: Plots of the training and validation loss obtained with different augmentation factors.