A SENSOR FUSION APPROACH FOR AUTONOMOUS DRIVING

(OTONOM SÜRÜŞ İÇİN SENSÖR BİRLEŞTİRME YAKLAŞIMI)

by

Bekir Eren ÇALDIRAN, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

July 2022

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Prof. Dr. Tankut ACARMAN for his support and guidance in a compelling time due to pandemic. The availability of Prof. Acarman every day of the week and every hour of the day has eased the process a lot.

My special thanks are to my family and especially my mother for their continuous support throughout the process of working from home.

I also would like to thank to my fellow from Galatasaray University, who is graduating at the same time with me by accomplishing her Doctoral thesis, my elder sister Ezgi ÇALDIRAN, for setting up a great example in the family.

July 2022
Bekir Eren ÇALDIRAN

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Human mistakes constitute very much of the car crashes and over the entire world, annual death rate of traffic accidents is over a million. Besides from the environmental and social contributions of autonomous cars, the potential to lower human risk factor in driving will result with less traffic accidents happening. To drive, autonomous vehicles need to understand the environment around. Multi-class objects including stationary and dynamic objects, vulnerable road users, cars, pedestrians, traffic lights, drivable areas and lane lines should be detectable for an intelligent vehicle. Humans use their eyes and brain to interpret the driving environment. For autonomous vehicles to interpret the environment, neural networks as brains and different sensors that have the capability of human's eye, or even sensors with further abilities than an eye are needed.

When considered individually, all sensors used in perception has their advantages and disadvantages. When camera and Lidar sensors are examined, both have drawbacks on changing environment, weather and illumination situations. Camera is disadvantageous in sudden illumination changing situations. Blinding effects of exposure to sudden light changes, entering or leaving a tunnel, driving at a dark, snowy, foggy road or even the sunlight hitting camera directly leaves the sensor unreliable for seconds. Lidar sensor on the other hand is light resistant. Its performance does not get effected by sudden illumination changes and it is advantageous as it can calculate the distance of the objects accurately. Yet it's an expensive sensor with low resolution and does not have the visual semantics capability of camera to classify the objects around. Sensor fusion is needed to create a system that has greater capability then single sensor setups, it is needed to overcome shortcomings of the sensors by relying on a combination of them. In this thesis, a sensor fusion approach is proposed to fuse outputs achieved from deep learning models that work on camera and Lidar sensors.

For camera sensor, a deep learning model, a multitask network is proposed to localize dynamic traffic objects such as cars, motorcycles, bicycles, buses, trucks, static traffic objects such as color classified traffic lights and traffic signs, pedestrians, segment drivable area and detect lane lines. This proposed multi-network achieves multiple tasks by itself faster and in real-time as normally there would be separate network models for each of these tasks working on parallel.

The multi-network proposed is trained and tested with Berkeley Deep Drive 100K dataset. Evaluation results show that the proposed method is the fastest multi-network on the dataset with 47.62 FPS. Around multi-networks, proposed work has the second place on drivable area segmentation and lane line detection. Dynamic object localization performance of the network is state-of-the-art with 40% performance increase compared to other models.

After being able to acquire knowledge about the environment by a single camera multi-network model, to further enhance the perception system, the information is fused with methods that work on Lidar. For Lidar sensor, two different modalities with capability to detect objects and ground plane respectively are used. Finally, the outputs of the two sensors are fused by proposed fusion algorithms and results are evaluated with KITTI dataset. The proposed fusion approach exceeds the performance of Lidar-only methods up to 9.87% category wise. Comparison with two different fusion approaches also show our superior performance.

Keywords: Deep learning, multitask learning, dynamic and static traffic object localization, pedestrian localization, traffic light classification, drivable area segmentation, lane line detection, lidar ground plane detection, lidar 3D object detection, sensor fusion, autonomous vehicles, intelligent transportation systems.

ÖZET

Trafik kazalarının çoğunu insan hataları oluşturur ve dünya genelinde trafik kazalarında yıllık ölüm oranı bir milyonun üzerindedir. Otonom araçların çevresel ve sosyal katkılarının yanı sıra, sürüşteki insan risk faktörünü azaltma potansiyeli, daha az trafik kazası meydana gelmesine olanak sağlar. Otonom araçların belirli bir rotada ilerleyebilmeleri için çevreyi anlamaları gerekir. Akıllı bir araç için sabit ve dinamik nesneler, savunmasız yol kullanıcıları, arabalar, yayalar, trafik ışıkları, sürülebilir alanlar ve şerit çizgileri dahil olmak üzere birden çok sınıfa ait nesneler algılanabilir olmalıdır. İnsanlar sürüş ortamını yorumlamak için gözlerini ve beyinlerini kullanırlar. Otonom araçlar için de bu yorumlamanın gerçekleşebilmesi için beyin olarak çalışacak sinir ağlarına ve insan gözünün yeteneğine sahip farklı sensörlere, hatta aracın bir insana kıyasla daha üstün sürüş kabiliyetine sahip olması isteniyorsa, bir göze göre daha ileri yeteneklere sahip sensörlere ihtiyaç vardır.

Çevreyi algılamada kullanılan sensörler ayrı ayrı ele alındığında, her birinin ayrı avantajı ve dezavantajı vardır. Kamera ve Lidar sensörleri incelendiğinde, değişen ortam, hava ve aydınlatma durumlarında ikisinin de dezavantajları vardır. Kamera ani ışık değişimlerine karşı dayanıksızdır. Tünele girer veya çıkarken, karanlık, karlı veya sisli bir yolda ilerlerken, ve kameraya doğrudan güneş ışığı gelirken kamera ani körleşir ve saniyeler boyunca güvenilmez hale gelir. Lidar sensörü ise ışığa karşı dayanıklıdır. Performansı ani ışık değişimlerinden etkilenmez ve etrafındaki nesnelerin mesafesini hesaplayabiliyor olması avantajlıdır. Yine de hem pahalı hem de düşük çözünürlüklü bir sensördür. Kamera gibi etrafındaki nesneleri sınıflandırmak için gereken görsel kabiliyete sahip değildir. Sensör füzyonu, tek sensörlü sistemlere göre daha fazla kapasiteye sahip bir sistem oluşturmak için gereklidir. Farklı sensörlerin kombinasyonu ile oluşan sistem sensörlerin bireysel eksiklerinin üstesinden gelir. Bu tezde, kamera ve Lidar sensörleri üzerinde çalışan derin öğrenme modellerinden elde edilen çıktıları birleştirmek için bir sensör

birleştirme yaklaşımı önerilmiştir. Kamera sensörü için; araba, motosiklet, bisiklet, otobüs, kamyon gibi dinamik trafik nesnelerini, rengi tahmin edilebilen trafik ışığı ve trafik levhası gibi statik trafik nesnelerini, yayaları, sürülebilir alanları ve şerit çizgilerini tespit edebilecek çok görevli bir derin öğrenme modeli önerilmiştir. Önerilen çok görevli model, normalde paralel olarak bu görevlerin her biri için ayrı ayrı çalışacak birden çok modelin yerine geçer ve bu işleri daha hızlı şekilde gerçek zamanda tek başına yapar. Önerilen çok görevli model, Berkeley Deep Drive 100K veri seti ile eğitilmiş ve test edilmiştir. Sonuç değerlendirmelerine göre, önerilen yöntem bu veri setindeki en hızlı çok görevli modeldir ve saniyede ortalama 47.62 kare işler. Çok görevli modeller arasında, sürülebilir alan ve şerit çizgileri tespitinde ikinci sırada yer almaktadır. Modelin dinamik nesne tespiti performansı diğer modellere kıyasla %40 performans artışı ile en iyi sonucu verir.

Tek bir kamera çoklu ağ modelinden çevre hakkında bilgiler edindikten sonra, algı sistemini daha da geliştirmek için elde edilen bilgi Lidar üzerinde çalışan yöntemlerle birleştirilmiştir. Lidar sensörü için nesneleri ve yer düzlemini algılama yeteneğine sahip iki farklı modalite kullanılmıştır. İki sensörün çıktıları önerilen füzyon algoritmaları ile birleştirilmiş, sonuçlar KITTI veri seti ile değerlendirilmiştir. Önerilen füzyon yaklaşımı, yalnızca Lidar ile çalışan yöntemlerin performansını kategori bazında %9.87'ye kadar arttırmıştır. Bunun üzerine iki farklı füzyon yaklaşımı ile yapılan karşılaştırmada da, bu tezde önerilen füzyon yaklaşımının performans üstünlüğünü gösterilmiştir.

Anahtar Kelimeler: Derin öğrenme, çoklu görev öğrenme, dinamik ve statik trafik objelerinin tanımlanması, yaya tanıma, trafik ışığı renklerinin sınıflandırılması, yol üzerindeki sürülebilir alanın bölümlenmesi, şerit çizgilerinin tespiti, lidar yer düzlemi tespiti, lidar 3 boyutlu obje tespiti, sensör birleşimi, otonom araçlar, akıllı ulaşım sistemleri.

1. INTRODUCTION

Intelligent and autonomous vehicles are not only needed to ease the task of transportation or reducing fuel consumption, carbon emission, but it is also needed to prevent human mistakes while driving. According to the Global Status Report published by the World Health Organization (WHO., 2018), the reported number of annual road traffic deaths reached 1.35 million in 2018, making it the world's eighth leading cause of unnatural death among people of all ages. Also, according to (Singh., 2015), 94% of serious crashes are due to human error. Autonomous cars can reduce the risk of crashes by limiting dangerous driver behaviors and human mistakes while driving.

Intelligent vehicle technologies, autonomous driving requires accurate and reliable perception to execute maneuvering tasks. Vehicles need to understand the driving environment to navigate safely. For an intelligent vehicle to detect multi-class objects including stationary and dynamic objects, cars, pedestrians, drivable area, a variety of technical features need to be engineered. To accomplish these tasks, humans use a combination of sensors, ears, eyes and brain to process. Therefore, to drive like a human, neural networks as brains, various sensors as eyes, ears or even maybe to drive better than a human, sensors that go beyond the limit of human sensing capabilities are needed for autonomous vehicles. Camera, Lidar and Radar are the mostly used sensors for perception. These sensors have limitations, drawbacks in changing environment situations.

Table 1.1 shows the performance comparison of the sensors on different situations. Changing weather conditions, rain, snow and fog leads these sensors to perform badly by harming visibility distance and range measurements. Besides from the sensor disadvantages on different weather and illumination conditions, unexpected behaviors of other drivers increase the complexity of the environment. To overcome each other's

disadvantages, as shown in the “Fusion” category on Table 1.1, sensor fusion is needed. With sensor fusion, the system has greater capability of sensing the environment by not only relying on a single sensor but a stronger combination of different sensors as each has its own advantages and disadvantages. In sensor fusion, a challenge is to answer, how to fuse information from different sensors since the output semantics and ranges are dissimilar. First, before fusing the information from multiple sources, positioning of the sensors are important for precise performance. Calibration here is used to calculate the relative position of each sensor and it is required to be able to use the overlapping data as information from the sensors will be fused together. Although calibration is a hard task, multi-sensor datasets provide already calibrated sources and the calibration files which saves time and effort of the researcher. After getting already calibrated input files from multiple sensors, it is only about the method, about to decide how the fusion is to be made.

Table 1.1: Comparison of the commonly employed sensors based on technical characteristics and other external factors. (Yeong et al., 2021).

Factors	**Camera**	**LiDAR**	**Radar**	**Fusion**
Range	+	+	++	++
Resolution	++	+	**X**	++
Distance Accuracy	+	++	++	++
Velocity	+	**X**	++	++
Color Perception, e.g., traffic lights	++	**X**	**X**	++
Object Detection	+	++	++	++
Object Classification	++	+	**X**	++
Lane Detection	++	**X**	**X**	++
Obstacle Edge Detection	++	++	**X**	++
Illumination Conditions	**X**	++	++	++
Weather Conditions	**X**	+	++	++

There are three sensor fusion approaches, early fusion, mid-level fusion and late fusion. Late fusion fuses the high-level representations, results from different applied methods and combines the outputs into one result. Mid-level fusion uses descriptions, features from different sensors and operate on the combined result. Early fusion completely fuses the raw data from different sensors at the lowest level possible. Late fusion approaches are less complex and require less communication between the sensors since the fusion is on output level. Mid-level fusion approaches are harder to train and require large amount of data. With this approach different features from the sensors can be combined to increase the performance. For early fusion approaches, each sensors advantages are combined in the lowest level resulting in a more precise data and a faster fusion performance. The disadvantage is that fusing direct outputs from multiple sensors in real-time requires a strong communication and memory systems. In this work, to accomplish sensor fusion, two sensors, camera and lidar with a late fusion approach is used.

To interpret the environment with camera sensor, multiple tasks need to be accomplished. Normally there are different deep learning models to handle different tasks. Models of single task object detection, traffic light classification, drivable area segmentation and lane line detection working on parallel would achieve the environment interpretation traditionally. Recently, multi-task networks are proposed to handle different tasks altogether. With the right configurations and training, multi-task networks are capable of having better performance than single task networks on both accuracy and speed. With multitask networks, multiple tasks are trained in the same model at the same time. In preparation of this thesis, to make use of the camera sensor, we proposed our own deep learning model, a multitask network that is able to interpret driving environment by itself. Proposed model is able to localize dynamic traffic objects such as car, bike, motorcycle, bus and train, static traffic objects as traffic sign and color classified traffic light, pedestrians, drivable area and lane lines. We called this dynamic and static multi-network as “DynasticNet”. We refer to it as so throughout the rest of the thesis.

To train and test DynasticNet, we used Berkeley Deep Drive 100K dataset (BDD100K) (Yu et al., 2018). BDD100K is one of the most complex driving datasets proposed. The reason is that compared to other datasets, on all its tasks, drivable area segmentation, lane line detection, semantic segmentation, object detection, BDD100K is the most challenging one. BDD100K training dataset has a lot of occluded, truncated and nighttime images which makes it harder to learn. Images in this dataset are collected in different locations with crowdsourcing and the variety of environment conditions in the dataset makes DynasticNet even more prepared to real world applications.

We tested each task of our multi-network separately with different performance metrics, and real-world application performance is tested with random videos from different countries of the world, on different weather conditions, day and night times of the day. Videos are gathered online and not related to training dataset.

To interpret the environment with Lidar sensor, two different modalities with capability to detect objects and ground plane respectively are used. After having both camera and Lidar sensor inputs interpreted with a modality to achieve its task, proposed fusion methods are used to create a vector space that has both results fused accurately. True positive results from the camera frame and 3D lidar image are acquired to eliminate false results. The proposed approach assures improvement of the performance with respect to a single-sensor modality. In fusion part, we used KITTI dataset (Geiger et al., 2012) that provides synchronized camera, lidar frames and the calibration files. Rest of the work is organized as follows: In the next section, literature review on camera and lidar perception models and some fusion approaches are given, then the methodology is elaborated for firstly DynasticNet, and the fusion methods, followed by experimental studies and benchmarked results. Finally, some conclusions are given.

2. LITERATURE REVIEW

In this section, first camera perception models of single and multi-task are reviewed. Then review for perception models of lidar and fusion methods are given. For camera models, in the open literature, most methods handle tasks separately. Single task networks for lane detection, drivable area segmentation and traffic object detection are reviewed respectively. In following multi-task networks are detailed. For fusion part, first single modality lidar 3D object detectors are reviewed followed by the review of camera lidar fusion models.

2.1 Lane line detection

Earlier methods for lane line detection used machine learning algorithms to detect lane lines. The methods used had a poor and slow performance. Then with the rising application of deep learning, network models specified for lane line detection tasks are proposed. To exemplify the single task lane line detection networks, some network models are given.

Spatial Convolutional Neural Network (SCNN) in (Pan et al., 2018) makes use of convolutional layers to enable messages pass between pixels across rows and columns in a layer, but it is considered slow for a single task network due to slice-by-slice convolution. When a faster model is considered, in (Qin et al., 2020) a global feature is introduced by a row-based classification by locating lane lines on certain rows. In (Hou et al., 2019), SAD-ENet, a lightweight network that uses Self Attention Distillation (SAD) to allow model to learn from itself without additional supervision or labels are proposed. Examined models are used in the lane line detection performance comparisons. These single task models are specified for only lane line detection tasks.

2.2 Traffic object detection

Recently, the performance of object detection models is improved with great momentum. There have been different model proposals, one stage methods, two stage methods, detections with anchors, etc. On top of that as classification models used as backbones are also enhanced, with newer models such as, ResNet in (He et al., 2016), EfficientNet in (Tan & Le, 2019) and RegNet in (Radosavovic et al., 2020), object detectors gain a speed and performance boost. Example object detection models are given in the following. One stage methods such as SSD in (Liu et al., 2016) and YOLO in (Redmon et al., 2016) combine classification and localization. Based on these networks, there are also detection networks proposed on top of them. Two stage methods use region proposal to extract features and on top of the proposals, objects are localized and classified. Region based Convolutional Neural Network (R-CNN) in (Girshick et al., 2014) and Region-based Fully Convolutional Network (R-FCN) in (Dai et al., 2016) are representatives for two stage detectors. In comparison with respect to two-stage methods, one stage methods are suitable for intelligent vehicles since they have the potential to be executed on a real-time basis.

2.3 Traffic object segmentation

Segmentation methods try to estimate each pixel of the image to classify it correctly. Compared to an object detection model, segmentation models have much more classes to handle. When examples of segmentation methods capable of detecting dynamic and static traffic objects and pedestrians are examined, Pyramid Scene Parsing Network (PSPNet) in (Zhao et al., 2017) exploits the capability of global context information by different region-based context aggregation through pyramid pooling module together with the proposed pyramid scene parsing network. Deeplabv3+ in (Chen et al., 2018) has a spatial pyramid pooling module and encode-decoder structure combined for the advantages from both methods. In (Yin et al., 2020), Disentangled Non-Local Neural Networks (DNLNet) uses a disentangled non-local block to decouple attention computation and visual clues of salient boundaries. These models mentioned are used in the performance evaluation of traffic object localization.

2.4 Drivable area segmentation

Drivable area is segmented under semantic segmentation tasks when deep learning methods are introduced. Under segmentation tasks, usually the road is segmented as a whole. The segmentation methods do not have the separation we have, which is to segment to road but also learn the drivable space on it, considering each lane, and upcoming traffic lane on a two-way drive road. When road segmentation via segmentation models is examined, in (Long et al., 2015) Fully Convolutional Networks (FCNs) have been proposed by making use of a Convolutional Neural Network (CNN) classifier by replacing fully connected layers with 1x1 convolutions. Then, transposed convolutions are utilized to up-sample back to the original image size. Unet in (Ronneberg et al., 2015) uses the encoder-decoder architecture and skip connections to obtain high resolution outputs. These models mentioned are used in our model structure and the usage details are given in the upcoming section.

2.5 Multitask learning

Rather than focusing on single task networks of computer vision, multi-task models deployed for simultaneously handling different tasks together to be advantageous on both performance, resource usage and speed. We focus multi-task models that perform detection of drivable area, lane line and traffic objects tasks, which have attracted a lot attention recently. In (Qian et al., 2020), DLT-Net capable of segmenting drivable area, lane line and detects traffic objects, has a common encoder and three decoder branches. Decoders in DLT-Net are connected with context tensors. Context tensors are based on the information from drivable area, and these tensors increase the performance by concatenating drivable area parameters with lane line and traffic object branch. The DLT-Net underlying methodology is based on the idea that all lane lines and some of the traffic objects (only vehicles in their case) are located on the drivable area. Our presented multinetwork, DynasticNet, detects traffic sign and lights that are above the drivable area. A context tensor is not obtained since we are also focused on traffic lights, pedestrians and traffic signs as well. MultiNet (Teichmann et al., 2016), has simultaneous street classification, road segmentation and traffic object detection for cars

with one encoder and three separate decoders. Cell based method is used in detection branch of MultiNet which detects object based on anchors. YoloP in (Wu et al., 2021) detects lane lane, drivable area and traffic objects. YoloP is on a real-time basis, uses the BDD100K dataset, and reaches a high-level result in accuracy and speed. MultiNet is trained with the KITTI dataset in (Geiger et al., 2012). DLT-Net, YoloP and DynasticNet has been trained with the BDD100K dataset in (Yu et al., 2018). While comparing the slight differences, KITTI dataset does not include lane line, and MultiNet does not have lane line detection task. Although DLT-Net includes lane line detection task, lane line evaluation results have not been presented. In consequence, DynasticNet's lane line evaluation results, cannot be compared with these existing multi-task networks except YoloP. DLT-Net and MultiNet segments drivable area as a whole road. In DynasticNet, our presented approach separates lanes while segmenting drivable area. When drivable area is segmented, complete road is not segmented as drivable area, instead each lane in the drivable area is separated and detected. And the opposite direction of the road is detected as a non-drivable area, which is well presented for autonomous driving and lane keeping. DLT-Net and YoloP uses Feature Pyramid Network in their encoder design in (Lin et al., 2017). After shrinking the image to a certain level, before giving feature maps to separate decoder branches, both approaches make use of FPN to fuse different feature map results. After the fusion of neighbor layers, output layers are given to the decoders.

2.6 Lidar-only 3D detection networks

Along with the introduction of new models, stand-alone 3D lidar detection performance has been gradually improved. 3D lidar detection models can be separated based on adapting one-stage and two-stage methods as in 2D detection models that operate on image. In (Lang et al., 2019), PointPillars is an early adaptation of one-stage methods. PointPillars projects the point cloud to bird's eye view maps then applies 2D CNN to achieve 3D detection. To present point clouds in vertical columns as pillars, PointPillars uses PointNet (Qi et al., 2017) in the encoder part. Two stage VoxelNet in (Zhou & Tuzel, 2018) divides point clouds to 3D voxels and applies 3D CNN to learn features. In (Yan et al., 2018), Second has added 3D sparse convolution to VoxelNet and achieved

faster and efficient processing of voxels. Then two-stage PV-RCNN in (Shi et al., 2020) has extended Second by adding key-points branch to preserve 3D structural information. PVRCNN uses advantages of both 3D voxel CNN and PointNet-based methods to learn high-quality 3D proposal generation and flexible receptive fields. Two-stage PointRCNN in (Shi et al., 2019) uses raw point clouds to generate 3D proposals. PointRCNN generates 3D proposal and then refines in second stage. Another two stage Part-A2Net learns points distribution of 3D objects to predict part location and refine 3D proposals in the second stage by using collected part locations (Shi et al., 2020). Lidar methods used reach at high level in performance using the KITTI dataset (Geiger et al., 2012).

2.7 Camera Lidar fusion methods

Single sensor methods of detection/segmentation on both camera and lidar are getting more promising in terms of performance metrics. Overall, it is still a question to answer one should use multiple sensors or single sensor setups to achieve these tasks. While single modality is acceptable, an integration of modalities to increase the performance at a higher level is an inevitable experiment. Problems of sensor fusion are then expensive prices of lidar, the calibration and synchronization issues and memory/latency problems. Even after overcoming the problems mentioned, performance gain is not certain. Most methods do not increase the performance of baseline lidar-only methods. Pointfusion presented in (Xu et al., 2018), Frustum PointNet in (Qi et al., 2018) and Frustum ConvNet in (Wang & Jia, 2019) is a 3D detector that makes use of 2D detectors to help canalize 3D processing space to a narrowed domain. Frustum PointNet, detects 2D boxes first, crops point cloud according to the boxes and applies PointNet in (Qi et al., 2017) to get 3D box estimation. Two stage MV3D in (Chen et al., 2017), AVOD in (Ku et al., 2018) extracts features from images and point cloud converted bird’s eye view maps. Then a fusion is achieved by converting 3D proposals to 2D feature maps. MMF in (Liang et al., 2019) predicts dense depth from LiDAR and image and uses the predicted depth points to find dense correspondences between the feature maps from the two sensor modalities and fusion method is a point-wise feature. CLOCs in (Pang et al., 2020) and Pointpainting in (Vora et al., 2020) are the two fusion methods that increase

the performance in comparison to other fusion methods. CLOCs fuses two single modality detection models dedicated to a camera and to a lidar, respectively. CLOCs operates on the combined output candidates before Non-Maximum Suppression (NMS) of any 2D and any 3D detector. CLOCs uses prediction scores of 2D detections from the camera and 3D detections from the lidar to achieve 3D detection. CLOCs improves the performance of baseline lidar only methods. Pointpainting makes use of 2D segmentation methods to increase the performance of 3D lidar only detection models. Pointpainting reflects segmentation results of 2D network on intensity channel of lidar input and re-train the baseline lidar-only models with painted point clouds. Then, 3D detection networks learn to detect painted points.

Our work is motivated by Pointpainting considering the reflection part of 2D segmentation results on point clouds. Instead of re-training, we combine the results of our 2D multi-network or any 2D segmentation network with any 3D lidar detection network to achieve a higher level in 3D detection and drivable area segmentation performance. We operate on output level, bringing multiple outputs together to create a stronger vector space. As we operate on output level of different sensor modalities, our method is a late fusion approach. We process each modality separately and fuse in decision level. Since we do late fusion, we do not create new detections but correct false positives.

3. METHODOLOGY

In this section, two different methodologies of this work are given. First chapter concerns the methodology for building the 2D camera multi-network DynasticNet. The process of creating a multi network with multiple output branches and multiple loss functions are explained in detail. Then the methodology for sensor fusion is given. In sensor fusion part, first, how 3D boxes and ground plane results are achieved from 3D lidar point cloud inputs are explained. Later, the processes of bringing the camera and Lidar results on the same vector space is explained. Lastly, two different fusion methods proposed to fuse results of traffic objects/3D bounding boxes and drivable area/ground plane are explained.

3.1 Methodology of DynasticNet

DynasticNet is presented in Figure 3.1. Encoder branch of the network shrinks the image and gets feature maps with the help of pre-trained VGG-16 architecture, consisting of 3x3 convolutional and max pooling layers. The 3x3 convolutional layers learn to extract feature maps from input images, max-pooling layers use these feature maps to create smaller images with extracted features remaining. With the help of transfer learning, as VGG-16 network is pre-trained with ImageNet (Russakovsky et al., 2015) dataset, which nearly has 1000 classes, using its already trained parameters to achieve feature extraction is advantageous and thanks to pre-trained parameters of VGG16. The last layer of the encoder branch is given directly as an input to three decoders. For the first two up sampling operation inside each decoder branch, the corresponding layer from the encoder is fused. DynasticNet in Figure 3.1 can jointly detect lane lines, drivable area and localize traffic objects as dynamic, static and

pedestrians. If the static object is a traffic light, the color is also classified. In the decoder branches, the proposed method in FCN in (Long et al., 2015), increasing the size 8 times in the last layers (output layers with stride 8) is used to give results with enhanced details. To further enhance the resolution of our results and the performance, skip layers, which have been presented by Unet in (Ronneberg et al., 2015), are used as well.

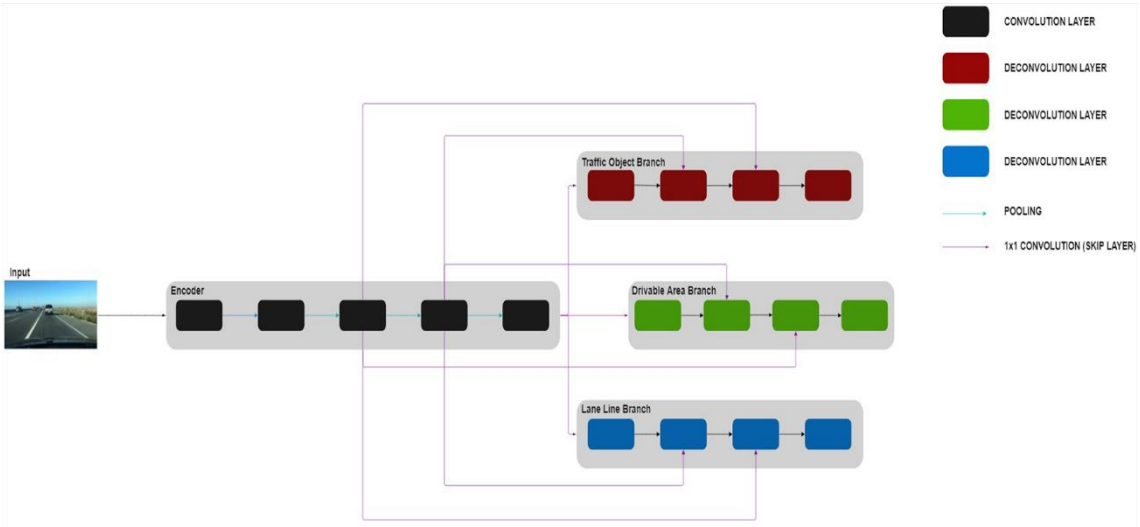


Figure 3.1: DynasticNet has one shared encoder for feature extraction and three decoders for three tasks. The last feature map is given as the input to decoders, the fusing operation is used in each decoder branch separately.

3.1.1 Encoder branch

A pre-trained network model, VGG16 in (Simonyan et al., 2014), is used as the feature extractor. Main advantage of using convolutional layers to extract features is to be able to extract high level features. Last fully connected layers of VGG are replaced with 1x1 convolutions to feed three decoders separately from single encoder output. Encoder output has the size of original image divided by 32.

3.1.2 Decoder branch

For all decoder branches, the same single output from encoder part is given as an input. Separate skip layers for each branch are used to fuse information from different layers. By fusing information with skip layers, high resolution outputs are achieved. In decoder branches, transposed convolutional layers are used instead of up sampling and then applying convolution. Last transpose convolution layers have a stride of 8, as originally used in FCN in (Long et al., 2015). Branches are elaborated as follows:

Drivable Area and Lane Line Branch: all branches get an input of original image 32 times divided. With skip connections and transposed convolutional layers, input image is up sampled to the original image size. The output image of this branches has two channels, which represent the probability of each pixel for drivable area and background, and also the probability of each pixel for lane line and background.

Traffic Object Branch: In this branch, multiple outputs, each can be considered as different detection result are fused together under three classes, which are so-called dynamic traffic objects, static traffic objects and pedestrians. This is not as equal to semantic segmentation methods that calculate each class as a different object with their unique label shapes. Instead, localization is applied here based on the bounding box labels where inside it is filled with class labels.

Loss Function and Training: from three decoders, the network gives three different outputs. For drivable area branch, cross entropy loss is used. For lane line and traffic object branch, class imbalance is a problem. Lane lines constitute a very few of the image compared to background area samples. Considering traffic object case, the number of dynamic objects is higher in comparison with the number of static objects whereas static objects constitute a large number with respect to the number traffic light colors. To tackle these class imbalances, a special approach, focal loss in (Lin et al., 2017) is used. Focal loss focuses on misclassified examples. It aims to minimize the classification error between pixels and focuses on hard labels. Focal loss adds a modulating factor to cross entropy loss with tunable focusing parameter which helps the

network to balance the weights, to not only focus on the most common class and to learn hard labels easier. Equation 3.1 and 3.2 gives binary cross entropy (CE) loss and focal loss (FL), respectively (Lin et al., 2017). In Equation 3.2, β indicates the weighting factor and μ indicates the tunable focusing parameter.

$$CE(p,y) = \begin{cases} -log(p) & if\ y = 1 \\ -log(1-p) & otherwise. \end{cases} \tag{3.1}$$

$$FL(p,y) = \begin{cases} -\beta(1-p)^{\mu} log(p) & if\ y = 1 \\ -(1-\beta)\, p^{\mu} log(1-p) & otherwise. \end{cases} \tag{3.2}$$

For all branches, each pixel in the image has a class. For drivable area and lane line branch, each pixel has either background or foreground value, either zero or one. For traffic object branch, each pixel has either background or one of the fifth foreground classes, either zero or a number between one and five.

The multi-task loss is given in Equation 3.3. Total loss is the sum of three loses coming from the three branches. *Lc* is short for cross entropy loss and *Lf* is for focal loss. Drivable area branch uses cross entropy loss, traffic object and lane line branches use focal loss. For three branches *ti*, *di*, *li* indicates network prediction probability value for traffic object, drivable area and lane line branch respectively and *ti**, *di**, *li** indicates the corresponding ground truth value. Since each pixel of the image has a loss value, the total loss of an image is calculated by the sum of all pixels, therefore W indicates the image width and H indicates the image height. *α1, α2 and α3* are the loss weights of traffic object, drivable area and lane line branch respectively. Weight of the losses are chosen to avoid focusing on most common mask on the image during the training stage. In consequence, lane lines, compared to drivable area, are subject to a much higher

weight as the ground truth masks covers a very few of the data content provided by the frame. These values are tuned to reach at the highest result extracted from our network. Weights are chosen as 1, 1 and 10, respectively. While training, Adam is used as the optimizer and learning rate is set to 1e-4.

$$Loss = \alpha 1 \sum_{k}^{W*H/2} Lf(ti, t_i{}^{*}) + \alpha 2 \sum_{k}^{W*H} Lc(di, d_i{}^{*}) + \alpha 3 \sum_{k}^{W*H} Lf(li, l_i{}^{*}) \quad (3.3)$$

For training and evaluation purposes, images from the BDD100K dataset are resized from 1280x720x3 to 640x384x3 to use computing resources in an efficient and effective manner, to save memory and speed up the learning process. The used encoder VGG16 is trained with ImageNet in (Russakovsky et al., 2015), with images having the size of 224x224. Therefore, training our network with images resized by half (1280x720 to 640x384) has increased the level in accuracy and computational speed. To ease the learning process, preprocessing is used. The preprocessing method used in the training of VGG16, which zero centers the values with respect to ImageNet dataset, has been applied. In training and evaluation, Tesla P100 was the GPU used.

3.2 Methodology of Sensor Fusion

The architecture is presented in Figure 3.2. Separate inputs coming from Camera and Lidar simultaneously are processed through the networks independently. The fusion method works with any Camera 2D segmentation and Lidar 3D object detection model. Apart from proposed DynasticNet, two segmentation networks BiseNetv2 (Yu et al., 2021) and Deeplabv3+ (Chen et al., 2018) are also utilized to provide diverse evaluation results. In parallel to camera feature extraction, 3D Lidar point cloud input is used with existing 3D object detectors to propose 3D bounding boxes. RANSAC algorithm in (Fischler & Bolles, 1981) is also used to extract ground plane from lidar input.

After the detections are achieved, Camera results are stacked to 3D Lidar space. To use 2D segmentation results on Lidar space, 2D results are converted to 3D points with the

help of calibration files provided by the KITTI dataset. The final Lidar vector space has segmentation labels from the camera for the coinciding area. Lidar points intensity channel are replaced by channel-wise activations from 2D network and result is a painted point cloud. Figure 3.2 shows a general block diagram representation of the proposed fusion method.

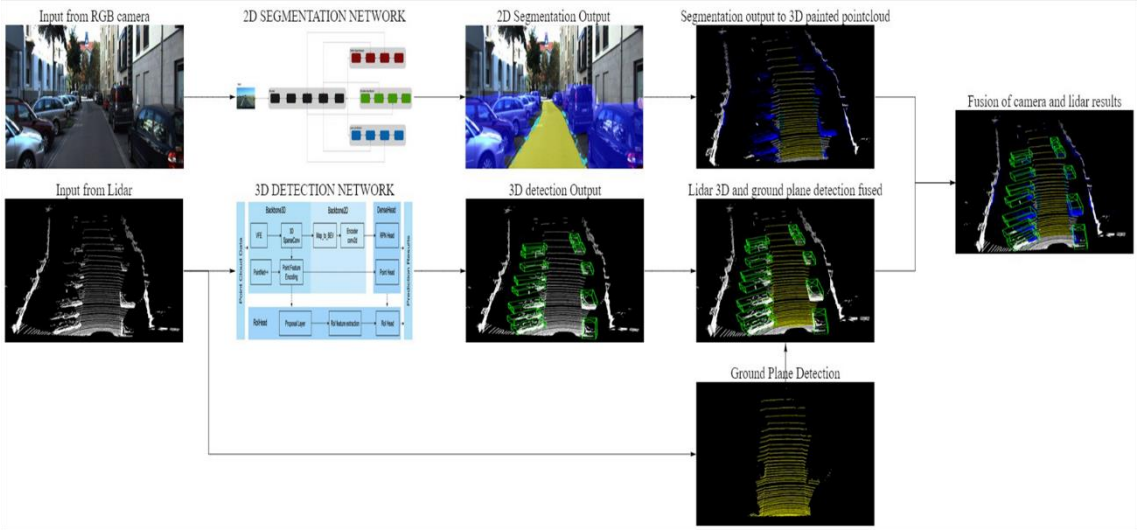


Figure 3.2: The presented architecture has standalone camera and lidar inputs. 2D output from any 2D-camera segmentation network is converted to 3D point clouds. 3D object detection results from 3D lidar object detection model is fused with lidar ground plane detection. Lidar-only result is then fused to 3D painted point cloud of segmentation result to generate final output.

3.2.1 Camera modality

To get segmentation results from camera input, any 2D segmentation network can be used. DynasticNet is used to interpret the driving environment scene in the evaluations. BiseNetv2 network is also implemented as a 2D camera segmentation method. BiseNetv2 is finetuned with the KITTI semantic segmentation dataset in (Alhaija et al., 2018) before usage. Lastly, Deeplabv3+ segmentation network trained with Cityscapes dataset in (Cordts et al., 2016) is implemented.

3.2.2 Lidar 3D modality

To ease the usage of multiple 3D detection networks and the process of testing, the method presented in (OpenPCDet Development Team, 2020) is implemented. They provide 6 different 3D object detection models with 9 different versions with testing and training configurations.

For Lidar ground plane extraction, the RANSAC algorithm is used. Since RANSAC is considered to be computationally slow the parameters are tuned to require less points to be processed and the number of iterations is reduced. Also, only the points below certain height within the lidar view is checked.

3.2.3 Fusion methods

After stacking results of both 3D bounding boxes and camera segmentation to the same vector space, algorithm given in Algorithm 3.1 is used. Inputs are bounding boxes proposed by the 3D lidar network, distance of the boxes on the longitudinal axis measured in meters, camera and lidar thresholds, lowered subjects to the increase in distance, and the bounding box prediction scores.

First, boxes are eliminated by checking if they are inside the camera range as boxes outside the camera field of view are not touched. To match with the area front camera sees, Lidar input is used on the longitudinal axis between 5 to 65 meters and -25 to 25 meters on the lateral axis.

By trial and error, bounding boxes are placed on each 10 meters' distance on the longitudinal axis to set threshold values as they depend on the distance. At each 10 meters, a lower threshold value is required to validate both lidar and camera input. Since this is an asymmetric fusion as bounding box proposals only come from one sensor (lidar in this study), the information provided by the camera is used to correct the false positive results. More explicitly, if certainty of 3D network decreases, the painted points by 2D segmentation output are used if its condition, threshold value is met. Camera

threshold is pointwise and threshold value depends on the painted points, which indicates whether the points inside the bounding box are painted and if so, how many is actually painted i.e., density. In case the threshold is not met, for instance although lidar network detected an object but the camera did not localize it, the detection is eliminated as it is now marked as a false positive.

Algorithm 1 *CameraLidarAgreement*(bxs, dx, ct, lt, pt)

Inputs :
bxs *is Boxes proposed by 3D lidar network*
dx *is Distance of the boxes on the longitudinal axis*
ct *is Camera 2D threshold*
lt *is Lidar 3D threshold*
pt *is Lidar detected box prediction score*

Output :
True positive **bxs**

Method :
Get **bxs** *within camera range.*
Group **bxs** *by* **dx**.
Set distinct **lt** *and* **ct** *on each group.*
If **pt** $<$ **lt** *check if* **ct** *is met*
If so set as true positive else false positive.

Algorithm 3.1: Fusion method for 3D bounding boxes and segmentation results.

The output of drivable area or road segmentation from camera is also fused with Lidar ground plane extraction. This fusion method corrects false positive drivable area results as camera can provide unreliable results at poor lightning conditions. Some false positive drivable area result examples are observed such as detection of roadblocks, sideways, grass, tunnel walls and ceiling of bridges. To fuse the camera drivable area and lidar ground plane, the method shown in Algorithm 3.2 is used. Inputs are drivable area and ground plane. Drivable area is projected onto ground plane and only joint points are retrieved.

Algorithm 2 *CameraLidarDrivableAreaAgreement*(da, gp)

Inputs :
da *is Drivable area proposed by 2D network*
gp *is Ground plane from 3D lidar input*

Output :
True positive **da**

Method :
Project **da** *onto* **gp**
Remove non − intersection points.

Algorithm 3.2: Fusion method for camera drivable area and lidar ground plane detection.

4. EXPERIMENTAL STUDY

In this section as before, first the experiments of DynasticNet are given, then the experimental study for sensor fusion is detailed. Different configurations for evaluations and performance metrics are given for each section. Performance metrics are given with tables and bar charts.

4.1 DynasticNet experiments

DynasticNet is trained end to end with the BDD100K dataset in (Yu et al., 2018). Since the label of the test set is not published, network is evaluated with the validation set. The experiments are conducted starting with drivable area, for lane lines and traffic objects.

4.1.1 Drivable area results

Mean intersection over union (mIoU) is used to evaluate drivable area results. For drivable area, BDD100K separates drivable areas as alternative and direct lanes. In DynasticNet, we do not have that separation. We only separate background and drivable area with all lanes having the same category. In drivable area, each lane is same for us without considering if it is an alternative or direct lane. Throughout our evaluation study, the mIoU metric is performed on seprating background and foreground (drivable area). In Table 4.1, drivable area segmentation results are compared with multi-task networks. There are other single-task segmentation methods retrained with BDD100K that have drivable area segmentation but separates lanes as direct and alternative then calculate IoU for both classes and average. Mean mIoU of these methods are in the interval 77% and 85% according to results published in official BDD100K model zoo. When segmentation of drivable area without the distinction of direct and alternative

lanes are checked, DynasticNet has the second place among multi-networks using the BDD100K dataset. When all methods compared, while constructing YOLOP, they retrained PSPNet to compare its performance with their own work. Although official PSPNet drivable area segmentation mIoU result is 83.80, it is stated that YOLOP got mIoU result of 89.6 by retraining, ranking the metric result of DynasticNet in the third place considering drivable area segmentation around all BDD100K models. DynasticNet outperforms MultiNet and DLT-Net by 22.1% and 21.2%, respectively. In terms of speed, DynasticNet model is 5 times faster than these models and it has the first place in inference speed by exceeding YOLOP 16.15%. Visualization results of the drivable area segmentation are plotted between Figure 4.1 and Figure 4.3.

Table 4.1: Benchmarking results of drivable area segmentation

Networks	**mIoU(%)**	**Speed (fps)**
MultiNet	71.60	8.60
DLTNet	72.10	9.30
YOLOP	91.50	41.00
DynasticNet	87.39	47.62

Figure 4.1: Visualization of the drivable area segmentation result. In two-way road case, incoming lane is not segmented.

Figure 4.2: Visualization of the drivable area segmentation result. Each lane is separated in terms of drivable area.

Figure 4.3: Another visualization of the drivable area segmentation results that shows each lane is separated in terms of drivable area.

4.1.2 Lane line results

Lane lines are trained and evaluated based on the actual lane masks from BDD100K. For the evaluation of lane line detection, to only focus on foreground, Intersection over union (IoU) metric for lanes are used. Also, pixel accuracy is calculated for this task. Firstly, the model is trained while preserving the input size 1280x720x3 and lane line masks. Hence, effects of input size and preprocessing on both inference speed and accuracy are compared. Then, same cases are trained with input size 1280x720 and 640x384 with and without preprocessing. In training of both 1280x720 images, lane line masks are untouched. For training with 640x384 input size, lane line pixel widths are increased to 2 and effect of preprocessing is evaluated. As VGG16 is trained with

images with size 224x224, which is much smaller compared to 1280x720, resizing the image size by half increases the performance and inference speed. VGG16 is trained with the ImageNet dataset, therefore pre-processing images properly according to the preprocessing technique used in the training of VGG16 increases the performance. After lowering image size by half and saving memory, performance gain is inevitable as both batch size and total amount of image given to the network increased. The effect of increasing the lane pixel widths positively affected the training performance. The experiment results are given in Table 4.2, In Table 4.2, pp denotes preprocessing, * indicates our latest and official model, which is the most efficient, and ~ indicates "without" in this case. mIoU shows drivable area performance.

Table 4.2: The effect of frame size and lane line pixel width to performance metrics

Input Size	**mIoU (%)**	**Lane IoU (%)**	**Speed (fps)**
1280x720 ~pp	72.10	7.10	24.93
1280x720 with pp	75.80	8.70	24.93
640x384 ~pp	81.10	11.30	47.62
*640x384 with pp	87.39	15.15	47.62

In Table 4.3, results for lane line detection are compared versus the existing models. For lane line detection, only multi-task network that has evaluation is YOLOP. For this task, comparison is with YOLOP and other single task state-of the-art lane detection. All other methods in this comparison use single line (usually center line) for a lane line representative. We use several lane classes and since we trained the network with official BDD100K masks, it is more challenging to learn. BDD100K is designed for lane instance classification and indicates a lane line with two lines. In terms of *IoU* for lanes, since we train with official masks, it is more complex for the network to represent a lane line with two lines. In terms of accuracy, a higher result is obtained. In SCNN,

input size is 800 × 288 and pixel sizes for lane lines are 16 in training and 30 in evaluation. These changes make their evaluation very specific to lane lines. ENet-SAD and YOLOP uses images with size 640x384 and single center line with width set to 8 in training and 2 at evaluation. For us, pixel width is 2 in training and testing also the image size is same with ENet-SAD and YOLOP, which is 640x384. Around multi-networks, DynasticNet has the second place on lane line detection. Visualizations for lane line detection are given in Figure 4.4, Figure 4.5 and Figure 4.6.

Table 4.3: Benchmarking results of lane line detection

Networks	**Accuracy (%)**	**IoU(%)**
SCNN	35.79	15.84
ENet-SAD	36.56	16.02
YOLOP	70.50	26.20
DynasticNet	70.27	15.15

Figure 4.4: Visualization of the lane line detection result. Road curbs and lane lines are detected. These are the direct masks from predictions and are not post processed by any means of line fitting.

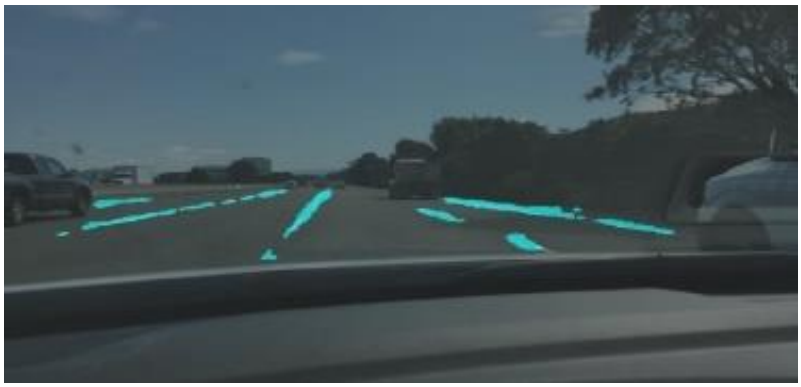


Figure 4.5: Visualization of the lane line detection result. Dashed lines are detected.

Figure 4.6: Another visualization of the lane line detection result.

4.1.3 Traffic object localization results

For traffic objecs, MultiNet, DLT-Net and YOLOP have a detection task and can only be evaluated by vehicle objects. Localization is used in our work and Intersection over union (IoU) scores for each class can be calculated based on the ground truth bounding box labels of the objects where the bounding box is filled with class labels like in semantic segmentation. In semantic segmentation, shapes of the labels are different for cars, people, bike etc., Here, all classes have box as the label shape. Using a common shape for labeling helps us bring multiple classes under the same category and learn all of them as one. Instead of creating an object detector, as our aim is to work with multiple tasks, this localization technique of objects as dynamic/static objects are useful and faster. To create a detector for these tasks, instead of grouping multiple classes

under super categories, each class would be considered separately. Pedestrians and traffic signs would be two of the different classes. Also red light and green light would be classified beforehand. Here instead, a practical way is used and the probability of dynamic objects, static objects, pedestrians, traffic light colors, their multi-class localization on image is achieved. To calculate the performance, Intersection over union (IoU) metric is used. Each corresponding ground truth and network output result for a class is mapped together to get their intersection and union. Then intersection of these results is divided by their union to achieve IoU. The performance is compared with single task semantic segmentation methods re-trained with the BDD100K dataset. The IoU metric results of specific objects given in BDD100K model zoo are used to get performance results of the segmentation methods. While calculating IoU for dynamic objects, the mean of other methods IoU results on car, bus, bicyle, motorcycle and truck are calculated. Since traffic lights are classified in this work but not in segmentation methods, separate categories used to calculate the performances. Considering dynamic objects' IoU, segmentation methods "motorcycle" and "truck" category results lower their mean. Since the data size of red light and pedestrians is lower compared to other categories in the training data, our results are lower on these categories as expected. Mean IoU (mIoU) is calculated according to the categories on Table 4.4.

Table 4.4: Results for traffic object localization. Comparison results versus the state-of-the-art single task segmentation methods on BDD100K dataset.

Networks	**Dynamic IoU (%)**	**Pedestrian IoU (%)**	**Sign IoU (%)**	**Light IoU (%)**	**Red Light IoU (%)**	**Green Light IoU (%)**	**mIoU (%)**
DNLNet	53.12	53.92	49.23	40.60	-	-	49.22
PSPNet	52.66	55.08	48.98	41.52	-	-	49.56
DeepLab v3+	52.06	55.28	49.71	42.06	-	-	49.78
Dynastic Net	73.54	41.38	42.88	-	31.58	41.51	46.18

As cars, bike, motorcycle, bus, truck is grouped and trained together under dynamic traffic objects, a performance boost is gained instead of focusing each of these objects separately. Although segmentation methods are one of the greatest models, they do not perform that great at BDD100K dataset due to its complexity. The presented approach performs better on green lights because of the imbalance between red and green light. There are lots of images in the dataset, captured at nighttime. Nighttime images make it complicated to separate red lights from car's red stopping light, causing misclassification of a red traffic light. And secondly, depending on the state regulations, lights are not mounted in a unique manner, some lights are mounted with a metal pole holding the light and some are mounted with a cable above the road surface. Traffic object localization results are visualized in Figures between 4.7 and 4.9.

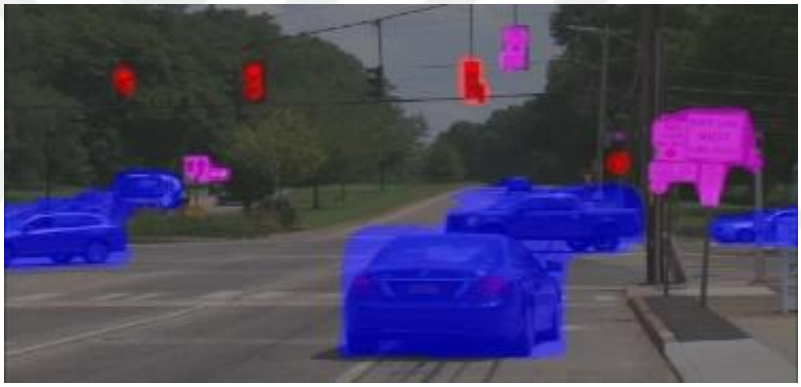


Figure 4.7: Visualization of the traffic object localization result. Red color indicates red light, dark purple indicates traffic signs and blue indicates dynamic objects.

Figure 4.8: Visualization of the traffic object localization result. Green color indicates green light, dark purple indicates traffic signs and blue indicates dynamic objects.

Figure 4.9: Another visualization of the traffic object localization result. Red color indicates red light, dark purple indicates traffic signs, magenta indicates pedestrians and blue indicates dynamic objects.

4.2 Sensor fusion results

This late asymmetric fusion method builds a strong vector space, contributes to the localization performance by fusing a camera and a lidar sensor. False positive results of 3D Lidar detectors are corrected with the help of camera localization. Cluster of parasitic lidar points to be misclassified and detected as the object are prevented. On one hand, camera is advantageous in such case to help lidar understand the semantic scene. On the other hand, lidar is advantageous to perform detections in low visibility and dark illumination conditions. In our experiments, lidar-only methods are tested using the default KITTI point cloud inputs. Boxes outside camera range are not processed, only the ones within the camera field of view are corrected. Since there is no ground truth label of lidar drivable area on KITTI, performance of 3D object detectors is given. For all tables in this section, blue color indicates the highest performance in category, green implies a performance improvement and red indicates a performance decrease. Table 4.5 shows results of lidar-only detectors and the fusion method proposed. Considering car class, our fusion method increases the performance of both bird's eye view and 3D detection tasks with all lidar networks. Performance gain is up to 2.08% achieved with Prcnn-IoU lidar-only model. On average all methods have finer performance. In Table 4.6, results for the pedestrian class are presented. High performance gains are reached for both bird's eye view and 3D detection tasks, performance gains are up to 9.87% with Second-IoU model.

Table 4.5: Fusion method applied to lidar based object detectors. Evaluation is with the KITTI validation set (40 recall metric). Bird's-eye view (BEV) and 3D detection average precision results are given for the car category.

Method	BEV Easy (%)	BEV Mod. (%)	BEV Hard (%)	3D Easy (%)	3D Mod. (%)	3D Hard (%)
PointPillars	92.04	88.06	86.66	87.75	78.40	75.18
PointPillars+DynasticNet	**92.76**	**89.03**	**87.10**	**88.47**	**79.28**	**75.55**
PointPillars+BiseNetv2	**92.71**	**89.02**	**87.20**	**88.42**	**79.27**	**75.57**
PointPillars+Deeplabv3+	**92.69**	**88.93**	**87.14**	**88.41**	**79.19**	**75.55**
Second	92.42	88.56	87.65	90.55	81.61	78.61
Second+DynasticNet	92.26	**89.50**	**89.53**	**91.49**	81.49	**78.98**
Second+BiseNetv2	**93.13**	**89.47**	**89.53**	**91.40**	**82.43**	**78.97**
Second+Deeplabv3+	**93.11**	**89.42**	**89.44**	**91.39**	**81.99**	**78.91**
PointRCNN	95.22	89.16	89.90	91.98	80.82	78.30
PointRCNN+DynasticNet	**95.54**	**89.81**	**89.92**	**91.98**	**81.20**	**78.36**
PointRCNN+BiseNetv2	**95.51**	**89.70**	89.49	**92.10**	**81.17**	**78.34**
PointRCNN+Deeplabv3+	94.62	**89.59**	89.22	**92.45**	**81.52**	**78.33**
PV-RCNN	93.02	90.33	88.53	92.10	84.36	82.48
PV-RCNN+DynasticNet	**93.47**	**90.54**	**88.72**	**92.56**	**84.58**	**82.68**
PV-RCNN+BiseNetv2	**93.46**	**90.55**	**88.75**	**92.55**	**84.55**	**82.65**
PV-RCNN+Deeplabv3+	**93.44**	**90.51**	**90.04**	**92.53**	**84.55**	**82.64**
PartA2-Free	92.84	88.15	86.16	91.68	80.31	78.10
PartA2-Free+DynasticNet	**93.46**	**89.06**	**87.87**	**92.29**	**81.10**	**79.63**
PartA2-Free+BiseNetv2	**93.43**	**89.14**	**87.92**	**92.21**	**81.17**	**79.67**
PartA2-Free+Deeplabv3+	**93.42**	**88.84**	**87.90**	**92.20**	**81.12**	**78.31**
VoxelRCNN	93.55	91.18	88.92	92.16	85.01	82.48
VoxelRCNN+DynasticNet	**93.72**	**91.74**	**90.24**	**92.41**	**85.03**	**82.53**
VoxelRCNN+BiseNetv2	**93.79**	**91.87**	**90.36**	**92.44**	**85.13**	**82.63**
VoxelRCNN+Deeplabv3+	**93.79**	**91.63**	**90.29**	**92.43**	**85.07**	**82.58**
Second-IoU	90.23	86.61	86.37	86.77	79.24	77.17
Second-IoU+DynasticNet	**91.06**	**87.60**	**88.26**	**87.63**	**80.20**	**77.44**
Second-IoU+BiseNetv2	**91.02**	**87.66**	**88.33**	**87.61**	**80.26**	**77.51**
Second-IoU+Deeplabv3+	**90.97**	**87.54**	**88.15**	**87.53**	**80.12**	**77.48**
Prcnn-IoU	95.07	88.85	86.77	92.01	80.61	78.33
Prcnn-IoU+DynasticNet	**95.35**	**89.53**	**88.25**	91.98	**80.69**	**78.51**
Prcnn-IoU+BiseNetv2	**95.33**	**89.68**	**88.24**	**92.24**	**82.40**	**78.59**
Prcnn-IoU+Deeplabv3+	**95.70**	**89.63**	**88.23**	**92.36**	**81.84**	**80.41**
PartA2-Anchor	92.90	90.06	88.35	92.15	82.91	82.00
PartA2-Anch+DynasticNet	**93.62**	**90.42**	**90.07**	**92.71**	**83.71**	**82.40**
PartA2-Anchor+BiseNetv2	**93.55**	**90.42**	**90.14**	**92.70**	**83.80**	**82.38**
PartA2-Anch.+Deeplabv3+	**93.53**	**90.38**	**90.09**	**92.68**	**83.72**	**82.28**

In Table 4.7, results for the cyclist class are given. In cyclist category, high performance gains are reached. Performance gain is up to 3.28% achieved with PointRCNN model.

Table 4.6: Fusion method applied to lidar based object detectors. Evaluation is with the KITTI validation set (40 recall metric). Bird's-eye view (BEV) and 3D detection average precision results are given for the pedestrian category.

Method	BEV Easy (%)	BEV Mod. (%)	BEV Hard (%)	3D Easy (%)	3D Mod. (%)	3D Hard (%)
PointPillars	61.60	56.01	52.04	57.30	51.41	46.87
PointPillars+Bisenetv2	**66.49**	**59.75**	**54.46**	**62.14**	**54.93**	**49.73**
PointPillars+Deeplabv3+	**65.88**	**60.60**	**55.79**	**61.45**	**55.32**	**50.55**
Second	60.73	56.57	52.14	55.94	51.14	46.16
Second+Bisenetv2	**66.81**	**58.91**	**52.84**	**60.65**	**53.15**	**47.07**
Second+Deeplabv3+	**64.79**	**59.91**	**55.07**	**59.39**	**53.95**	**48.16**
PointRCNN	66.32	58.31	52.68	62.75	55.12	48.33
PointRCNN+Bisenetv2	**68.86**	**58.94**	52.40	**65.12**	**55.84**	**49.05**
PointRCNN+Deeplabv3+	**68.03**	**59.70**	**52.72**	62.60	**55.80**	**48.74**
PV-RCNN	65.94	58.52	54.13	62.71	54.49	49.88
PV-RCNN+Bisenetv2	**70.47**	**61.76**	**56.25**	**67.27**	**58.38**	**53.14**
PV-RCNN+Deeplabv3+	**69.40**	**62.37**	**56.49**	**66.07**	**58.01**	**53.03**
PartA2-Free	75.48	69.84	**64.50**	72.36	66.40	60.07
PartA2-Free+Bisenetv2	**76.47**	**70.00**	63.71	**74.14**	66.27	**60.49**
PartA2-Free+Deeplabv3+	**76.03**	**70.07**	63.97	**72.86**	**66.41**	**60.82**
Second-IoU	40.65	37.63	35.36	36.45	33.58	31.17
Second-IoU+Bisenetv2	**50.52**	**45.81**	**42.32**	**44.97**	**41.31**	**37.33**
Second-IoU+Deeplabv3+	**48.89**	**44.97**	**42.02**	**43.42**	**39.74**	**36.93**
Prcnn-IoU	66.30	58.23	52.60	62.14	55.55	49.11
Prcnn-IoU+Bisenetv2	**68.50**	**59.43**	**52.72**	**65.72**	**57.10**	**49.82**
Prcnn-IoU+Deeplabv3+	**68.41**	**61.57**	**53.20**	**65.75**	**58.76**	**50.32**
PartA2-Anchor	70.53	64.20	59.25	66.89	59.68	54.62
PartA2-Anchor+Bisenetv2	**73.37**	**64.80**	58.91	**69.83**	**61.53**	**54.65**
PartA2-Anc.+Deeplabv3+	**72.59**	**65.37**	**60.16**	**69.05**	**61.81**	**55.49**

For Table 4.6 and 4.7, DynasticNet is not used as cyclist or rider classes are not used in the training. In Table 4.8, mean average precision results are given to show the average performance increases over all categories. General performance increase is up to 3.09% and 3.13% on BEV and 3D MAP respectively. Class-wise and average performance results are given with bar charts in Figure 4.10 and 4.11. Proposed fusion method at most have impact on pedestrian category followed by cyclist and car. Compared to other categories, pedestrians are the most disadvantaged in lidar cases since they get misclassified too easily with static objects such as the body of traffic sign or poles. Results in Figure 4.10 are from Pointpillars and Deeplabv3+ lidar and camera networks.

Table 4.7: KITTI validation set (40 recall). Bird's-eye view (BEV) and 3D detection average precision results are given for cyclist category.

Method	BEV Easy (%)	BEV Mod. (%)	BEV Hard (%)	3D Easy (%)	3D Mod. (%)	3D Hard (%)
PointPillars	85.26	66.24	62.22	81.57	62.81	58.83
PointPillars+Bisenetv2	86.90	67.38	62.77	83.01	63.87	59.57
PointPillars+Deeplabv3+	87.16	68.43	63.87	83.82	64.59	60.24
Second	88.05	71.16	66.89	82.96	66.74	62.78
Second+Bisenetv2	88.06	71.17	66.75	83.99	66.89	62.63
Second+Deeplabv3+	89.55	72.39	67.94	84.46	68.27	64.04
PointRCNN	92.72	74.25	69.72	89.06	69.50	66.16
PointRCNN+Bisenetv2	94.59	75.13	70.46	90.76	71.49	67.32
PointRCNN+Deeplabv3+	95.04	75.01	70.22	92.34	71.79	67.10
PV-RCNN	93.46	74.53	70.10	89.10	70.38	66.02
PV-RCNN+Bisenetv2	93.88	74.31	69.87	89.33	70.29	65.51
PV-RCNN+Deeplabv3+	93.97	74.78	70.14	89.78	70.40	65.85
PartA2-Free	93.23	78.50	73.93	91.92	75.33	70.57
PartA2-Free+Bisenetv2	94.47	78.13	73.43	92.95	74.83	71.15
PartA2-Free+Deeplabv3+	94.56	78.11	73.30	92.94	75.35	71.13
Second-IoU	86.45	67.73	63.39	83.19	63.75	60.24
Second-IoU+Bisenetv2	87.63	67.78	63.39	83.98	64.41	60.38
Second-IoU+Deeplabv3+	88.59	68.27	63.85	85.41	64.82	60.61
Prcnn-IoU	94.34	75.57	70.84	92.55	72.18	67.47
Prcnn-IoU+Bisenetv2	94.41	75.63	69.95	92.58	71.32	66.84
Prcnn-IoU+Deeplabv3+	94.35	76.41	71.72	92.63	73.09	68.46
PartA2-Anchor	91.96	74.64	70.63	90.34	70.14	66.93
PartA2-Anchor+Bisenetv2	92.23	74.64	69.61	90.44	70.47	66.12
PartA2-Anchor+Deeplabv3+	92.30	74.66	70.95	90.50	70.42	66.12

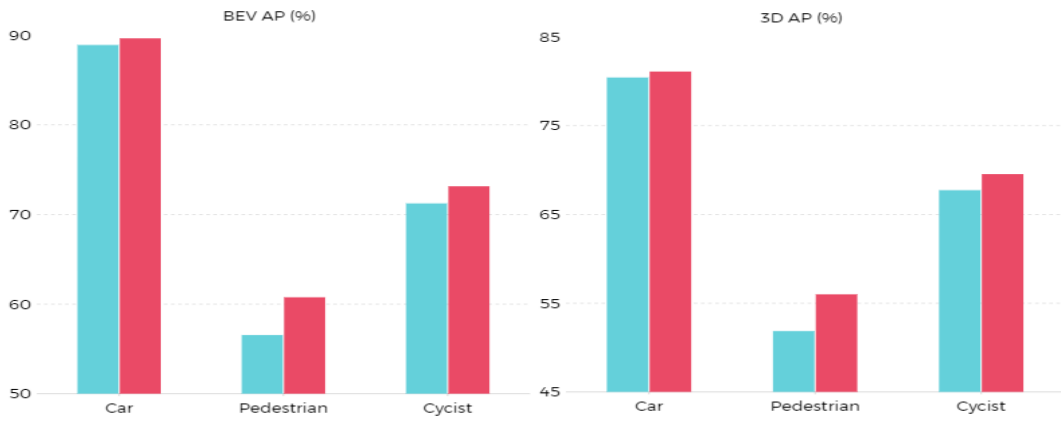


Figure 4.10: Bird's eye view (BEV) and 3D detection average precision results for different classes with KITTI validation set (40 recall).

Table 4.8: Bird's-eye view (BEV) and 3D detection mean average precision (mAP) results are given for the KITTI validation set (40 recall).

Method	BEV MAP (%)	3D MAP (%)
PointPillars	70.10	64.21
Fused Pillars	72.65	66.37
Difference	**+2.55**	**+2.16**
Second	72.10	66.50
Fused Second	73.91	68.07
Difference	**+1.81**	**+1.57**
PointRCNN	73.91	68.48
Fused PRCNN	74.77	69.70
Difference	**+0.86**	**+1.22**
PV-RCNN	74.46	69.74
Fused PvRCNN	75.89	71.07
Difference	**+1.43**	**+1.33**
PartA2-Free	78.83	74.01
Fused A2-free	**79.09**	**74.29**
Difference	**+0.26**	**+0.28**
Second-IoU	63.99	58.86
Fused Sec-iou	67.08	61.99
Difference	**+3.09**	**+3.13**
Prcnn-IoU	74.22	69.45
Fused Prcnn-iou	75.87	71.23
Difference	**+1.65**	**+1.78**
PartA2-Anchor	76.30	70.91
Fused A2-anchor	76.80	71.98
Difference	**+0.50**	**+1.07**

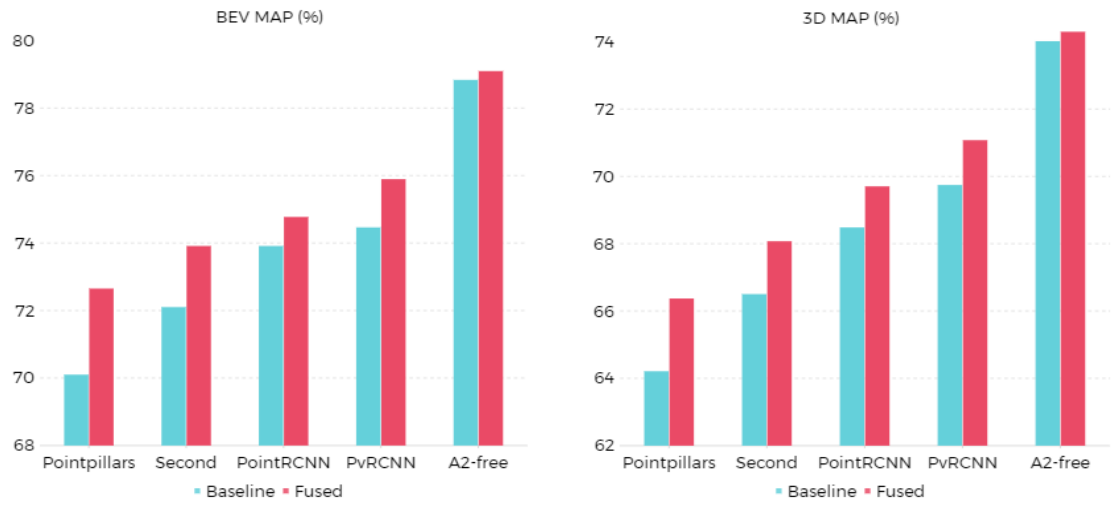
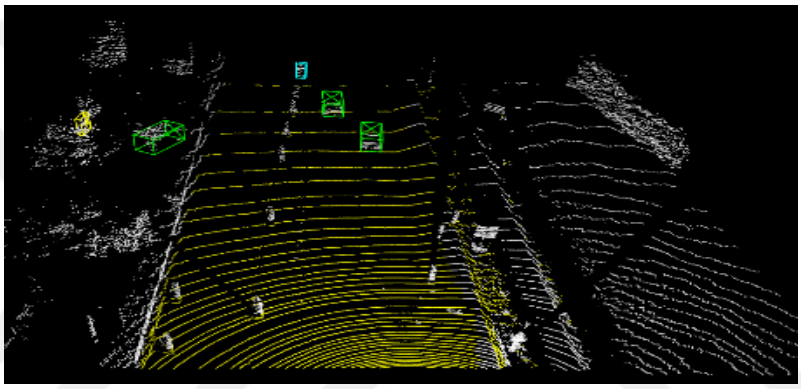


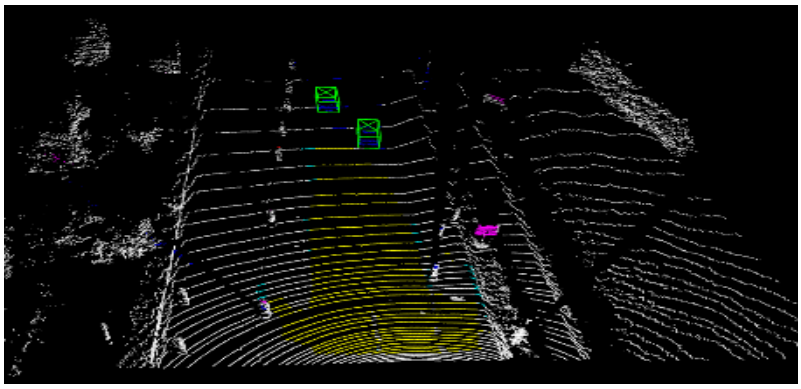
Figure 4.11: Bird's eye view (BEV) and 3D detection, mean average precision (MAP) results on KITTI validation set (40 recall).

For some objects, camera network can detect but lidar network falls back. This implies that both camera and lidar models should have high recall metrics. As the camera and lidar models perform better, higher performances are expected. Different combinations of threshold values and distance dependencies on fusion methods can change the performance accordingly. Fusion results of 2D multi-network DynasticNet and 3D lidar object detection network PartA2-free are visualized in different road scenarios between Figure 4.12 and 4.15. In figures, all corrected boxes from lidar results to fusion results (b to c in figures) are actual false positives. Different cases of hard scenarios and bad light conditions are shown.

(a) Camera 2D segmentation output.

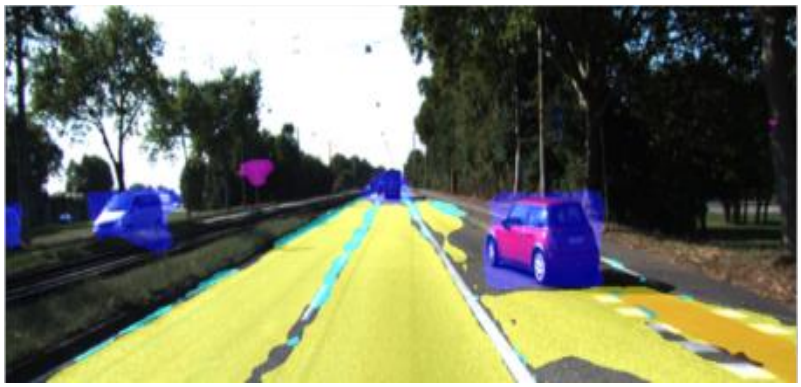


(b) Lidar 3D object and ground plane detection.

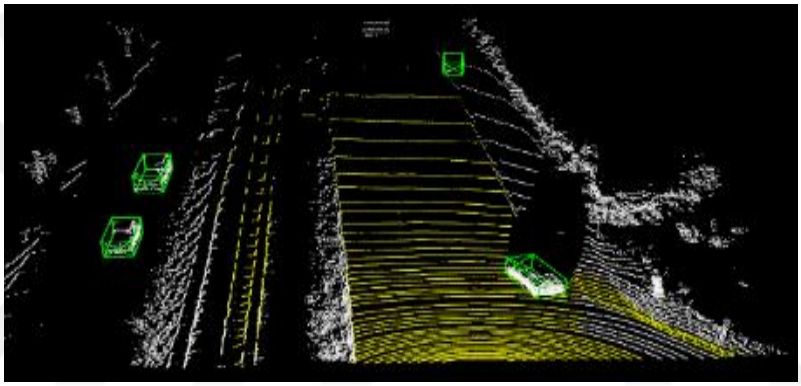


(c) Camera and lidar results fused.

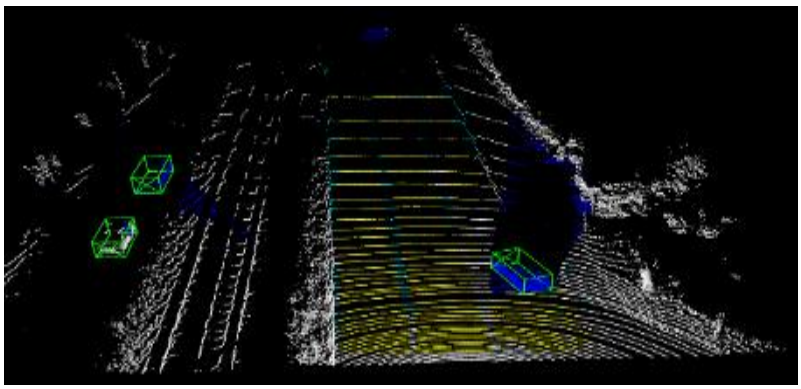
Figure 4.12: Results of the proposed fusion method. Boxes marked on the lidar output (b) that is eliminated in fusion results (c) is an actual false positive corrected by painted camera points. Pole detected as pedestrian and trees detected as car and cyclist by Lidar network are corrected by camera network.



(a) Camera 2D segmentation output.



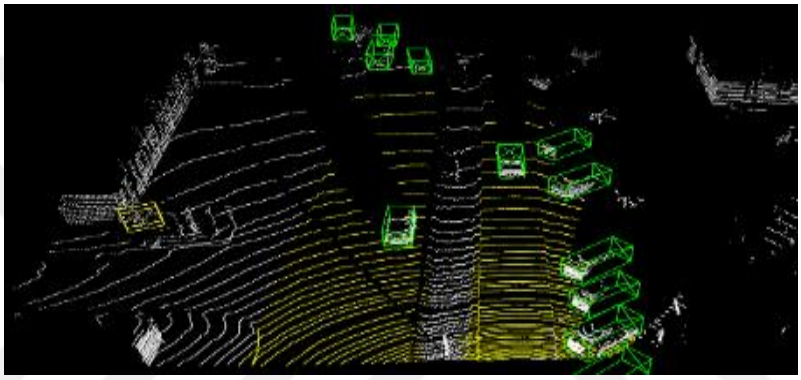
(b) Lidar 3D object and ground plane detection.



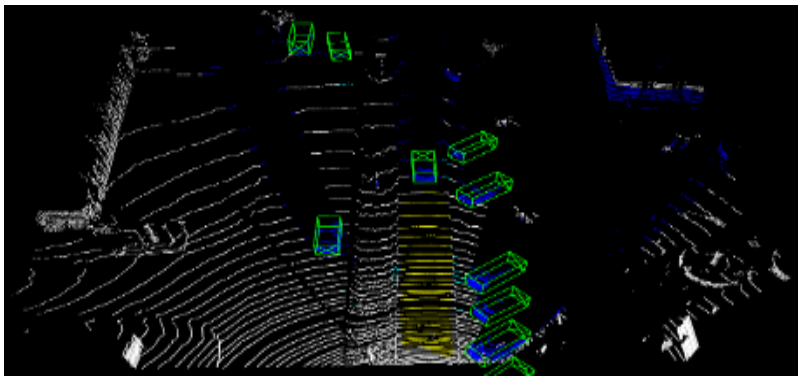
(c) Camera and lidar results fused.

Figure 4.13: Another fusion scenario, false positive result of trees detected as car by Lidar network is corrected by camera network.

(a) Camera 2D segmentation output.



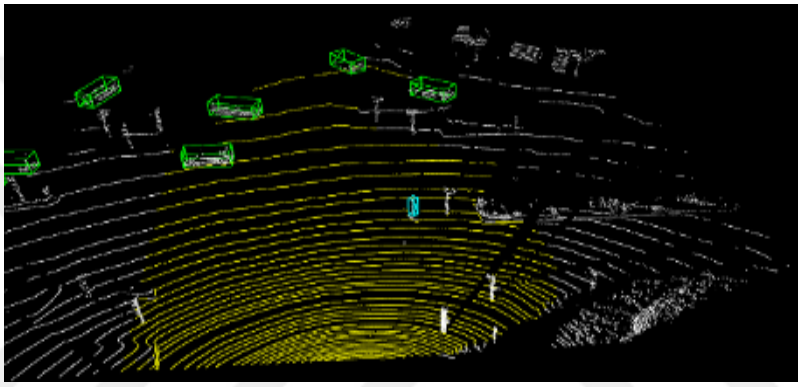
(b) Lidar 3D object and ground plane detection.



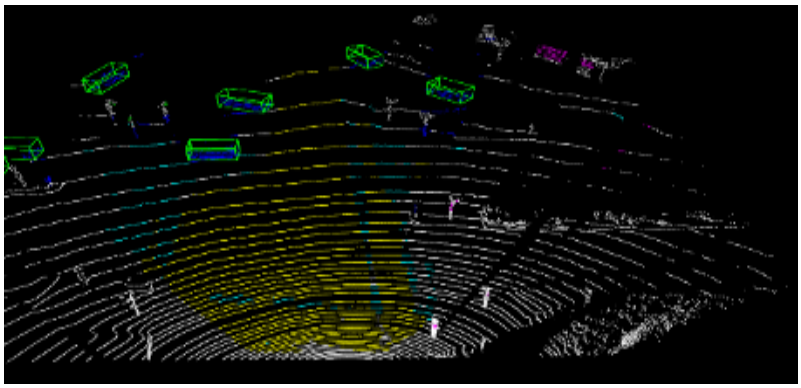
(c) Camera and lidar results fused.

Figure 4.14: Lidar ground plane (b) corrects camera false drivable area result (a). Misclassified grass by camera network is corrected by Lidar ground plane.

(a) Camera 2D segmentation output.



(b) Lidar 3D object and ground plane detection.



(c) Camera and lidar results fused.

Figure 4.15: Pole detected as pedestrian by Lidar network is corrected by camera network.

In Table 4.9, average latencies of the fusion methods are given. Since 2D segmentation and 3D lidar detection networks can be changed, different combinations can be made based on the tradeoff between performance and speed. We used RANSAC algorithm to extract Lidar ground plane, any method or 3D segmentation model can be used if ground plane is achieved as the output. As it is always required to convert 2D segmentation outputs to 3D lidar points to achieve the fusion, the latency of painting camera outputs to lidar points, 1.5 milliseconds are added necessarily. When necessary 1.5 milliseconds of conversion are added, only enhancing lidar 3D object detection bounding box outputs with painted points from camera segmentation results takes 4 milliseconds. If drivable area fusion is used by itself, the fusion costs 33.5 milliseconds. Comparably higher latency for camera drivable area and lidar ground fusion is due to high amounts of lidar points to be processed. When both bounding box and drivable area fusion methods are used together, the average latency is 36 milliseconds.

Table 4.9: Latency performance of applied fusion methods.

Method	**Latency**
Painting 3D Lidar Points using 2D segmentation output	1.5 ms
Camera-Lidar 3D detection fusion	2.5 ms
Camera-Lidar Drivable area fusion	32 ms

4.2.1 Comparison with a segmentation-based fusion

In this section, the performance of the proposed fusion method in PointPainting (Vora et al., 2020) is studied with camera segmentation BiseNetv2 network. PointPainting uses outputs from segmentation networks to re-train lidar only 3D detection networks. Retrained models therefore have semantic lidar points as inputs. For this comparison, 5 Baseline 3D lidar detection networks with masks from BiseNetv2 are trained, then also our proposed fusion method is used with retrained networks to evaluate the

collaboration of our fusion method with the re-training fusion method from PointPainting.

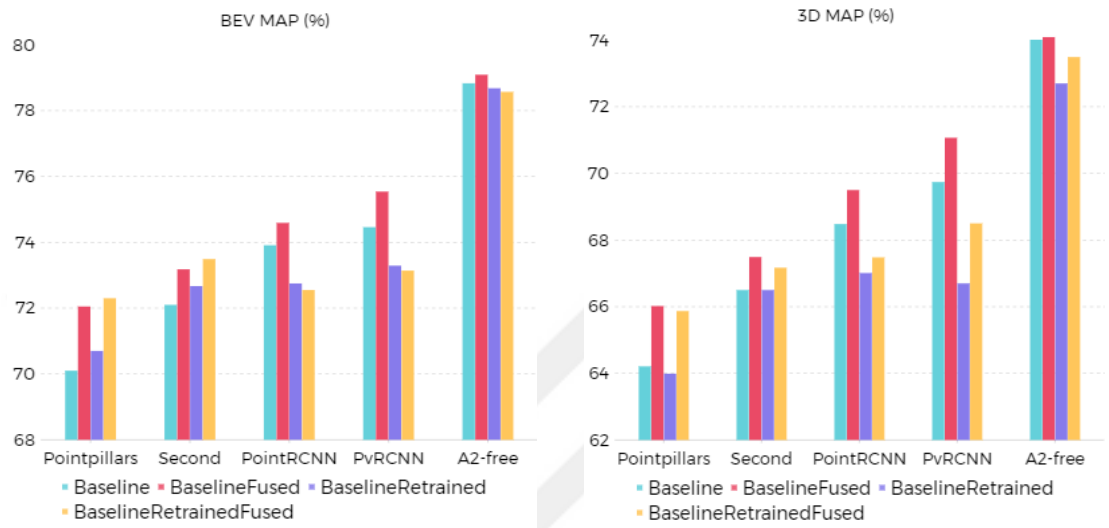


Figure 4.16: Bird's eye view (BEV) and 3D detection, mean average precision (MAP) results. Evaluation is done with lidar detection models on KITTI validation set 40 recall metrics.

In Figure 4.16, Baseline indicates the default lidar network models. BaselineFused indicates our fusion method. BaselineRetrained indicates the fusion method in PointPainting. Lastly, BaselineRetrainedFused indicates the collaboration the two fusion methods from PointPainting and ours. Only PointPillars and Second models have better performance on BEV MAP when used with PointPainting. In all cases our no-training required fusion method exceeds the performance increase of the retraining method from PointPainting and also the baseline models. Our method can run in parallel without the Lidar model depending on the camera output as the input, which creates a bottleneck. Pointpainting has this disadvantage, in order the re-trained lidar network to start detecting, an output from the camera segmentation network should be given as input.

4.2.2 Comparison with a detection based fusion method

In this section, the study is expanded with CLOCs in (Pang et al., 2020). Camera-LiDAR Object Candidates (CLOCs) fuses 2D and 3D detections by combining output candidates before Non-Maximum Suppression (NMS). Fusion method of CLOCs makes use of the probabilistic dependencies by confidence scores of detector networks. In this section official results from the published paper of CLOCs are used to compare the performance. To compare performance with CLOCs, 2D detection network MSCNN and 3D Lidar networks Pointpillars and Second are used as it is stated that only these networks published training configurations for pedestrian and cyclist categories. For our fusion method, Deeplabv3+ segmentation network is used. Comparison results are given in Figure 4.17.

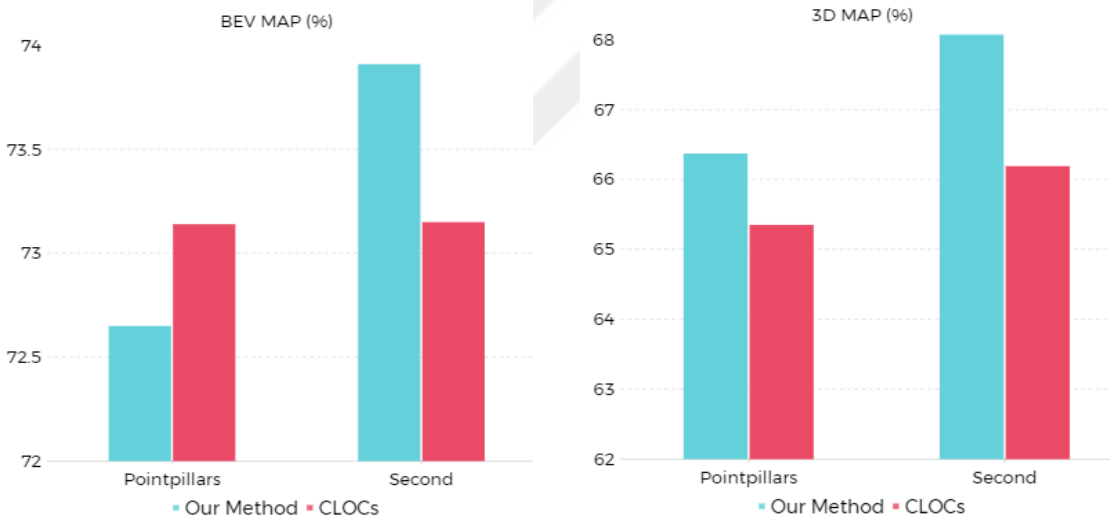


Figure 4.17: Bird's eye view (BEV) and 3D detection, mean average precision (MAP) results for all categories. Evaluation is done with KITTI validation set 40 recall metrics.

The proposed fusion method in CLOCs use 2D bounding boxes to support 3D boxes, the application of supporting 3D boxes with depth limited 2D boxes is disadvantageous as far as we concern. In 3D MAP results, our presented fusion approach exceeds CLOCs performance on both Lidar networks. For BEV MAP results, proposed method works better for the Lidar network "Second".

5. CONCLUSION

In this thesis, firstly for the camera part, a multi-network model, DynasticNet is proposed to handle different tasks simultaneously. Drivable area and lane line is detected while localizing dynamic objects, static objects, pedestrians and classified traffic light colors. DynasticNet has the second place on drivable area segmentation and lane line detection among multi-networks using the BDD100K dataset. In dynamic object localization, the presented multi-network model performs 40% better than the state-of-the art segmentation methods. By labeling different objects under the same class and training, classes with a smaller number of samples such as motorcycle, bike and truck are learnt in an efficient and effective manner.

Around multi-networks, DynasticNet has the first place for inference speed by 16.15%. Within average 21ms latency of the network, pre-processing, inference by model and post-processing latencies are combined, total time it takes to input an image and acquiring the output is given. By referring to our average 21ms latency, with 47.62 FPS, DynasticNet can run in real-time on standard and embedded devices.

For the Lidar combined sensor fusion part, an asymmetric late fusion approach is presented to create a sensor fusion model at the output/decision level. By fusing two different modalities each network's false positive results are eliminated and a superior vector space compared to single modality methods is created.

Results illustrate that the fusion method proposed increases the performance of Lidar only 3D detection networks. Considering 3D Lidar networks applied, the performance improvement is up to 9.87% class wise. The fusion method proposed can be used by any 2D segmentation and 3D Lidar detection network with no additional training required.

Considering runtime overhead, the fusion approach has low latency, improves the total performance for drivable area and 3D object detection bounding boxes below a processing delay of 36 milliseconds, 33 FPS at average. Without ground plane/drivable area fusion between lidar and camera, it only takes 4 milliseconds to enhance Lidar 3D bounding boxes with camera dynamic traffic object results.

REFERENCES

Alhaija, A. H., Mustikovela, S. K., Mescheder, L., Geiger, A., and Rother, C. (2018). Augmented reality meets computer vision: Efficient data generation for urban driving scenes., International Journal of Computer Vision, pp. 961-972.

Chen, X., Ma, H., Wan, J., Li, B. and Xia, T. (2017). Multi-view 3d object detection network for autonomous driving., Computer Vision and Pattern Recognition (CVRP), 2017 IEEE Conference on, IEEE, pp. 1907–1915.

Chen, L., Zhu, Y., Papandreou, G., Schroff, F. and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation., European conference on computer vision (ECCV), Springer, pp 801– 818.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. and Schiele, B. (2016), The Cityscapes dataset for semantic urban scene understanding., Computer Vision and Pattern Recognition (CVRP), 2016 IEEE Conference on, IEEE, pp. 3213-3223.

Dai, J., Li, Y., He, K., and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks., Advances in neural information processing systems, 29, pp. 379-387.

Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., Li, H. (2020). Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection., arXiv preprint arXiv:2012.15712 .

Fischler, M-A. and Bolles, R-C. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, communications of the ACM, volume 24, number 6, pp. 381-395.

Geiger, A., Lenz, P. and Urtasun, R. (2012). Are we ready for autonomous driving ? the kitti vision benchmark suite, Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, pp. 3354-3361.

Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation, Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE, pp. 580-587.

He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep residual learning for image recognition. Computer Vision and Pattern Recognition (CVRP), 2016 IEEE Conference on, IEEE, pp. 770-778.

Hou, Y., Ma, Z., Liu, C. and Loy, C. C. (2019). Learning lightweight lane detection cnns by self attention distillation, International conference on computer vision (IEEE/CVF), pp. 1013-1021.

Ku, J., Mozifian, M., Lee, J., Harakeh, A. and Waslander, S. L. (2018), Joint 3d proposal generation and object detection from view aggregation., International Conference on Intelligent Robots and Systems (IROS), pp. 1-8.

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J. and Beijbom, O. (2019). Pointpillars: Fast encoders for object detection from point clouds., Computer Vision and Pattern Recognition (CVPR), 2019 IEEE Conference on, IEEE, pp. 12697-12705.

Liang, M., Yang, B., Chen, Y., Hu, R. and Urtasun, R. (2019). Multi-task multi-sensor fusion for 3d object detection., Computer Vision and Pattern Recognition (CVPR), 2019 IEEE Conference on, IEEE, pp. 7345-7353.

Lin, T., Goyal, P., Girshick, R., He, K. and Dollár, P. (2017a). Focal loss for dense object detection., Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, IEEE, pp. 2980-2988.

Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S. (2017b). Feature pyramid networks for object detection., Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, IEEE, pp. 2117-2125.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y. and Berg, A. C. (2016). Ssd: Single shot multibox detector., European conference on computer vision, Springer, pp. 21-27.

Long, J., Shelhamer, E. and Darrell, T. (2015). Fully convolutional networks for semantic segmentation., Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on, IEEE, pp. 3431- 3440.

OpenPCDet Development Team, (2020), OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds.

Pan, X., Shi, J., Luo, P., Wang, X. and Tang, X. (2018). Spatial as deep: Spatial cnn for traffic scene understanding., AAAI Conference on Artificial Intelligence, Vol. 32, No. 1.

Pang, S., Morris, D. and Radha, H. (2020). CLOCs: Camera-LiDAR object candidates fusion for 3D object detection., International Conference on Intelligent Robots and Systems (IROS), pp. 10386-10393.

Qi, C. R., Su, H., Mo, K. and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation, Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, IEEE, pp. 652-660.

Qi, C. R., Liu, W., Wu, C., Su, H. and Guibas, L. J. (2018). Frustum pointnets for 3d object detection from rgb-d data, Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on, IEEE, pp. 918-927.

Qian, Y., Dolan, J-M. and Yang, M. (2020). Dlt-net: Joint detection of drivable areas, lane lines, and traffic objects, IEEE Transactions on Intelligent Transportation Systems, pp. 4670–4679.

Qin, Z., Wang, H. and Li, X. (2020). Ultra-fast structure-aware deep lane detection, European Conference on Computer Vision, Springer, pp. 276-291.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K. and Dollár, P. (2020). Designing network design spaces, Computer Vision and Pattern Recognition (CVPR), 2020 IEEE Conference on, IEEE, pp. 10428-10436.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016). You only look once: Unified, real-time object detection, Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on, IEEE, pp. 779-788.

Ronneberger, O., Fischer, P. and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation, International Conference on Medical image computing and computer-assisted intervention, Springer, pp. 234-241.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A-C. and Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge, International journal of computer vision, pp. 211-252.

Shi, S., Wang, X. and Li, H. (2019). Pointrcnn: 3d object proposal generation and detection from point cloud, Computer Vision and Pattern Recognition (CVPR), 2019 IEEE Conference on, IEEE, pp. 770-779.

Shi, S., Wang, Z., Shi, J., Wang, X. and Li, H. (2020a). From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network, IEEE transactions on pattern analysis and machine intelligence, pp. 2647-2664.

Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X. and Li, H. (2020b). Pv-rcnn: Point-voxel feature set abstraction for 3d object detection, Computer Vision and Pattern Recognition (CVPR), 2019 IEEE Conference on, IEEE, pp. 10529-10538.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

Singh, S. (2015). Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey.

Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks, International conference on machine learning, PMLR, pp. 6105-6114.

Teichmann, M., Weber, M., Zoellner, M., Cipolla, R. and Urtasun, R. (2018). Multinet: Real-time joint semantic reasoning for autonomous driving, IEEE Intelligent Vehicles Symposium (IV), pp. 1013-1020.

Vora, S., Lang, A. H., Helou, B. and Beijbom, O. (2020). Pointpainting: Sequential fusion for 3d object detection, Computer Vision and Pattern Recognition (CVPR), 2020 IEEE Conference on, IEEE, pp. 4604-4612.

Wang, Z. and Jia, K. (2019). Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection, International Conference on Intelligent Robots and Systems (IROS), pp. 1742-1749.

World Health Organization. (2018). Global Status Report on Road Safety, WHO: Geneva, Switzerland, ISBN 978-9241565684.

Wu, D., Liao, M., Zhang, W. and Wang, X. (2021). Yolop: You only look once for panoptic driving perception, arXiv preprint arXiv:2108.11250.

Xu, D., Anguelov, D. and Jain, A. (2018). Pointfusion: Deep sensor fusion for 3d bounding box estimation, Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on, IEEE, pp. 244-253.

Yan, Y., Mao, Y. and B. Li. (2018). Second: Sparsely embedded convolutional detection, Sensors, vol. 18, no. 10, pp. 3337, 2018.

Yin, M., Yao, Z., Cao, Y., Li, X., Zhang, Z., Lin, S. and Hu, H. (2020). Disentangled non-local neural networks, In European Conference on Computer Vision, Springer, pp. 191-207.

Yeong, D. J., Velasco-Hernandez, G., Barry, J., & Walsh, J. (2021). Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review. Sensors, 21(6), 2140.

Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V. and Darrell, T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning, Computer Vision and Pattern Recognition (CVPR), 2020 IEEE Conference on, IEEE, pp. 2636-2645.

Yu, C., Gao, C., Wang, J., Yu, G., Shen, C. and Sang, N. (2021). Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation, International Journal of Computer Vision, pp. 3051-3068.

Zhao, H., Shi, J., Qi, X., Wang, X. and Jia, J. (2017). Pyramid scene parsing network. Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, IEEE, pp. 2881-2890.

Zhou, Y.and Tuzel, O. (2018). Voxelnet: End-to-end learning for point cloud based 3d object detection, Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, IEEE, pp. 4490-4499.

BIOGRAPHICAL SKETCH

Bekir Eren Çaldıran started his high school education at Alibeyköy Anadolu High School in 2012. In his final year before college, he went to and graduated from Final Temel High School in 2016. Later, he got his bachelor's degree from Ozyegin University Computer Science Department in 2020. In 2022, he got his master's degree from Galatasaray University Computer Engineering Department. The master's thesis is presented with this document, and his supervisor was by Prof. Dr. Tankut Acarman.

So far, he has two publications. He had his first academic publication "Multi-network for joint detection of dynamic and static objects in a road scene captured by an RGB camera" at International Conference on Inventive Communication and Computational Technologies (ICICCT) in 2022. His second publication was "A Late Asymmetric Fusion Approach To Eliminate False Positives" at IEEE International Conference on Intelligent Transportation Systems (ITSC) in 2022.

His area of interest includes deep learning, autonomous vehicles, intelligent transportation systems and sensor fusion.

PUBLICATIONS

Caldiran, B.E. and Acarman, T. (2022). Multi-network for joint detection of dynamic and static objects in a road scene captured by an RGB camera, ICICCT, May 12-13, 2022, Namakkal, India, pp. 768 – 782.

Caldiran, B.E. and Acarman, T. (2022). A Late Asymmetric Fusion Approach To Eliminate False Positives, ITSC, October 8-12, 2022, Macau, China.