ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

MONODEPTH-BASED OBJECT DETECTION AND DEPTH SENSING FOR AUTONOMOUS VEHICLE VISION SYSTEMS

M.Sc. THESIS

Emre ÇETİN

Department of Computer Engineering

Computer Engineering Programme

JANUARY 2025

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

MONODEPTH-BASED OBJECT DETECTION AND DEPTH SENSING FOR AUTONOMOUS VEHICLE VISION SYSTEMS

M.Sc. THESIS

Emre ÇETİN
(504201518)

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Asst. Prof. Dr. Gökhan SEÇİNTİ

JANUARY 2025

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

MONODEPTH TABANLI OTONOM ARAÇ GÖRÜŞ SİSTEMLERİ İÇİN NESNE TESPİTİ VE DERİNLİK ALGILAMA

YÜKSEK LİSANS TEZİ

Emre ÇETİN
(504201518)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Asst. Prof. Dr. Gökhan SEÇİNTİ

OCAK 2025

Emre ÇETİN, a M.Sc. student of ITU Graduate School student ID 504201518 successfully defended the thesis entitled "MONODEPTH-BASED OBJECT DETECTION AND DEPTH SENSING FOR AUTONOMOUS VEHICLE VISION SYSTEMS", which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Asst. Prof. Dr. Gökhan SEÇİNTİ** Istanbul Technical University	.....
Jury Members :	**Prof. Berk CANBERK** Istanbul Technical University	.....
	Asst. Prof. Dr. Zafer DOĞAN Koc University	.....
		.....

Date of Submission : **26 December 2024**
Date of Defense : **22 January 2025**

To my parents, siblings and friends,

FOREWORD

I dedicate this thesis to my parents, siblings and friends. Their constant support has been invaluable. I am especially grateful to Özlem ER for her exceptional support.

I also extend my heartfelt thanks to my advisor, Assoc. Dr. Gökhan SEÇİNTİ, for his insightful guidance and expertise. Additionally, I am deeply appreciative of the examining committee members for their valuable time and contributions to my academic growth.

January 2025

Emre ÇETİN
Computer Engineer

TABLE OF CONTENTS

ABBREVIATIONS

CIoU	**:** Complete Intersection over Union
CNNs	**:** Convolutional Neural Networks
CSP	**:** Cross Stage Partial
DFL	**:** Distribution Focal Loss
DoF	**:** Degrees of Freedom
FPN	**:** Feature Pyramid Network
FPS	**:** Feature Per Second
IoT	**:** Internet of Things
IoU	**:** Intersection over Union
mAP	**:** Mean Average Precision
ONNX	**:** Open Neural Network Exchange
PAN	**:** Path Aggregation Network
RMSE	**:** Root Mean Squared Error
SSD	**:** Single Shot MultiBox Detector
SGM	**:** Semi-Global Matching
YOLO	**:** You Look Only Once

SYMBOLS

α : Weight assigned to the outer edges of the detected object
β : Weight assigned to the center points of the detected object
R : Weight average process ratios

LIST OF TABLES

Page

LIST OF FIGURES

MONODEPTH-BASED OBJECT DETECTION AND DEPTH SENSING FOR AUTONOMOUS VEHICLE VISION SYSTEMS

SUMMARY

This study explores object recognition and distance measurement technologies for autonomous vehicles and addresses advancements in this area. Autonomous vehicles are vehicles capable of perceiving objects in their surroundings and moving safely without human intervention. The progress in these technologies holds the potential to revolutionize mobility, largely driven by the significant role of artificial intelligence in enabling these advancements. Artificial intelligence is considered a fundamental component for autonomous vehicles to perceive environmental conditions and ensure safe travel. Techniques such as deep learning and machine learning allow vehicles to process data collected through cameras, lidars, radars, and other sensors to interpret their surroundings and provide safe responses.

The processes of object detection and depth estimation are continually being improved with traditional and AI-based methods.Deep learning models improve object detection and localization capabilities by processing complex image data, thereby enhancing the safety and performance of autonomous vehicles.

Autonomous vehicles use various sensors to detect environmental objects and determine distances. These sensors include lidars, radars, cameras, and ultrasonic sensors. Data collected from these sensors are processed to perceive surrounding objects and determine distances from the vehicle. While camera sensors are widely used, distance estimation with single-lens 2D cameras can be challenging.

Advancements in artificial intelligence techniques have achieved significant success with 2D camera images. Particularly, these techniques offer alternatives to challenges such as increasing sensor costs and the need for 3D cameras to adapt to various environmental conditions, enhancing the importance of progress in this area.

In this study, different artificial neural network models were used for object detection and depth estimation. Optimization efforts were conducted to adapt trained models for real-world applications, and testing on the Nvidia Jetson device was considered a crucial step. The developed application achieved successful results in real-time processing performance, representing a significant advancement in autonomous vehicle technologies.

The real-time processing performance of the developed application was determined to be 12 frames per second (FPS). Additionally, the average absolute error value obtained in the object detection and distance estimation areas was measured as 1.51 meters. These values indicate that the application operates quickly and reliably, accurately predicting object locations.

MONODEPTH TABANLI OTONOM ARAÇ GÖRÜŞ SİSTEMLERİ İÇİN NESNE TESPİTİ VE DERİNLİK ALGILAMA

ÖZET

Otonom araçlar, modern teknolojinin en dikkat çekici ve potansiyel dolu alanlarından birini oluşturuyor. Bu araçlar, insan müdahalesi olmadan çevrelerindeki nesneleri algılayabilen, çeşitli trafik koşullarında güvenli bir şekilde hareket edebilen ve hatta seyahat rotalarını planlayabilen araçlardır. Otonom araçların gelişimi, mobiliteyi kökten değiştirecek bir potansiyel taşıyor. Bu gelişimde, yapay zeka teknolojilerinin belirleyici bir rolü bulunmaktadır.

Yapay zeka, otonom araçların beyni olarak düşünülebilir. Derin öğrenme, makine öğrenimi ve benzeri teknikler, araçların çevrelerindeki nesneleri algılamalarını, bu verileri işlemelerini ve uygun tepkiler vermelerini sağlar. Örneğin, kameralar, lidarlar, radarlar ve diğer sensörler aracılığıyla toplanan verileri analiz ederek, yapay zeka sistemleri aracın çevresini anlamlandırır ve güvenli bir seyahat sağlamak için gerekli kararları alır.

Bu bağlamda, otonom araçların çevresel nesneleri algılayıp uzaklıklarını belirleme süreci, geleneksel ve yapay zeka tabanlı yöntemlerle giderek daha da geliştirilmektedir. Derin öğrenme modelleri, karmaşık görüntü verilerini işleyerek nesne algılama ve konum belirleme yeteneklerini artırabilir. Bu gelişmeler, otonom araçların güvenliğini ve etkinliğini artırmaya yönelik çabalarda önemli bir rol oynamaktadır.

Otonom araçlar, çevrelerindeki nesneleri algılamak ve konumlarını belirlemek için bir dizi sensör kullanır. Bu sensörler arasında lidarlar, radarlar, kameralar ve ultrasonik sensörler bulunabilir. Gelen veriler, aracın çevresindeki nesnelerin algılanması ve araçla nesneler arasındaki mesafelerin belirlenmesi için işlenir. Özellikle, kamera sensörleri, aracın çevresindeki nesneleri algılamak için yaygın olarak kullanılır. Stereo kamera sistemleri veya 3D kameralar kullanılarak elde edilen görüntüler, derinlik bilgisini sağlayarak çevredeki nesnelerin konumlarının belirlenmesine yardımcı olur. Ancak, tek mercekli 2D kameralardan mesafe tahmini yapmak zorlu bir süreçtir. Bu kameralar, derinlik algısını sağlamak için paralaks gibi stereoskopik bilgileri kullanamazlar, bu da mesafe tahmininin doğruluğunu etkiler.

Stereo veya 3D kameraların yanı sıra lidarlar ve radarlar gibi diğer sensörlerin kullanılması da otonom araçların nesneleri algılamasına ve mesafeleri belirlemesine yardımcı olabilir. Ancak, bu sensörlerin maliyeti genellikle yüksektir, bu da otonom araç teknolojisinin yaygınlaşmasını kısıtlayabilir.

Artan sensör maliyetleri ve 3D kameraların farklı ortam koşullarına uyum sağlama ihtiyacı, araştırmacıları 2D kamera görüntüleri üzerinde odaklanmaya yönlendirdi. Gelişen sinir ağları, bu 2D görüntüler üzerinde önemli başarılar elde etmeye başladı.

Bu başarılar, çeşitli yapay zeka teknikleriyle donatılan sinir ağlarının, çevresel nesnelerin algılanması ve mesafelerinin belirlenmesinde 2D kamera verilerini etkili bir şekilde kullanabildiğini göstermektedir. Bu gelişmeler, otonom araçlar ve diğer uygulamalarda kullanılan algılama ve mesafe belirleme sistemlerinin daha ekonomik ve esnek hale gelmesine katkı sağlayabilir. Dolayısıyla, 2D kamera görüntüleri üzerinde yapılan çalışmalar, artan sensör maliyetlerine ve 3D kameraların çeşitli ortam koşullarına uyum sağlama zorunluluğuna bir alternatif olarak öne çıkmaktadır.

Bu çalışmada, 2D kamera görüntülerinde nesne tespiti ve nesne mesafelerinin tahmini için iki ayrı yapay sinir ağı kullandık. Ayrıca, daha sağlam bir mesafe tahmini elde etmek amacıyla yeni bir algoritma olan Ağırlıklı Mesafe Tahmin Algoritması'nı uyguladık. Bu algoritma, nesnelerin tespit edilmesi ve konumlandırılması için kullanılan yapay sinir ağlarından elde edilen konum bilgilerini temel alarak daha doğru mesafe tahminleri sağlamayı amaçlamaktadır. Çalışmamız, 2D kamera görüntülerinin kullanımıyla ilgili olarak hem nesne tespiti hem de mesafe tahmini alanlarında yeni yöntemlerin geliştirilmesine katkıda bulunmaktadır. Bağımsız ve paralel olarak çalışan modellerde, nesne tespiti için You Only Look Once (YOLO) algoritmasını kullanırken, tahmin edilen mesafeleri algılamak için U-Net mimarisine dayalı Monodepth2 modelini kullanmaktayız. Yöntemin gerçek dünya koşullarında çalışmasını test etmek amacıyla Nvidia Jetson cihazı üzerinde test edilmesi önemlidir. Nvidia Jetson gibi yerel işlem gücüne sahip cihazlar, yapay zeka modellerini yerinde çalıştırarak gerçek zamanlı uygulamalarda yüksek performans sağlayabilen kompakt ve enerji-verimli sistemlerdir. Bu platformda test edilmesi, yöntemin mobilite ve pratik kullanılabilirliği açısından önemli bir adımdır. Nvidia Jetson üzerinde başarılı bir şekilde çalışması, yöntemin gerçek dünya uygulamalarında yaygın olarak kullanılabilirliğini ve güvenilirliğini artırır.

Nesne tespiti için YOLOv8 (You Only Look Once V8) algoritması tercih edildi. YOLOv8, görüntüdeki nesneleri tek bir aşamada algılayabilen ve sınıflandırabilen hızlı ve etkili bir derin öğrenme modelidir. Bu özelliği, gerçek zamanlı uygulamalarda kullanımını tercih edilen bir hale getirir. YOLOv8'in diğer geleneksel nesne tespit algoritmalarına kıyasla avantajı, tek bir geçişte nesneleri algılayıp sınıflandırması ve bu sayede yüksek hız ve etkinlik sağlamasıdır. Bu özellikler, yöntemin nesne tespiti sürecinde yüksek doğruluk ve hız sağlamak amacıyla tercih edilmesini sağlar.

Derinlik tahmini için Monodepth2 modeli tercih edildi. Monodepth2, U-Net mimarisi üzerine kurulmuş bir derin öğrenme modelidir ve tek bir görüntüden stereo eşdeğer derinlik haritaları oluşturmak için kullanılır. Bu model, tek kamera görüntülerinden derinlik tahmini yaparken yüksek doğruluk sağlar. U-Net mimarisi, girdi görüntüsünün her bir pikseli için derinlik bilgisi üretmek üzere tasarlanmış olup, özellikle tek gözlü kamera sistemlerinden elde edilen görüntüler için etkili bir şekilde çalışır. Monodepth2 modelinin tercih edilmesi, derinlik tahmini sürecinde yüksek doğruluk ve güvenilirlik sağlamak amacıyla yapılmıştır. Bu model, tek bir kameradan elde edilen görüntülerle nesne mesafelerini tahmin etmek için kullanıldığından, gerçek zamanlı uygulamalarda kullanımı yaygın olarak tercih edilen bir hale gelir.

Nesne algılamayla tanımlanan sınırlayıcı kutular içinde arka plan görüntüsü, genellikle algılanan nesnenin sınırlarının ötesine uzanır. Bu durumda, arka plana atfedilen pikseller, ortalama derinlik tahmininde bozulma yaratma potansiyeline sahiptir. Bu

durum, özellikle derinlik tahmini için tek bir görüntü kullanıldığında belirgin hale gelir. Bu endişeyi gidermek için, bir ağırlıklı mesafe tahmin algoritması kullanıyoruz. Bu algoritma, algılanan nesnelerin gerçek konumlarına daha yakın tahminler elde etmek için sınırlayıcı kutulardaki piksellerin ağırlıklı olarak dikkate alınmasını sağlar. Bu sayede, derinlik tahmininde olası bozulmalar azaltılarak daha doğru sonuçlar elde edilir.

Bu çalışmada, derinlik tahmini için KITTI Dataseti, nesne tespiti için ise BDD100K veri setleri kullanıldı. KITTI Dataseti, genellikle derin öğrenme modellerinin eğitimi ve değerlendirilmesi için kullanılan yaygın bir veri setidir ve genellikle otonom sürüş uygulamaları için kullanılır. Öte yandan, BDD100K veri seti, büyük ölçekli nesne algılama, sınıflandırma ve segmentasyon problemleri üzerine odaklanan bir veri setidir.

Eğitilen modellerin gerçek hayata uyarlanabilmesi için, optimizasyon çalışmaları gerçekleştirildi ve bu modellerin Nvidia Jetson cihazı üzerinde çalışabilirliği test edildi. Bu kapsamda, modellerin performansını artırmak ve daha etkin bir şekilde çalışmalarını sağlamak için optimizasyon adımları uygulandı. Ayrıca, modelin Nvidia Jetson gibi yerel işlem gücüne sahip cihazlarda doğru bir şekilde çalışıp çalışmadığını kontrol etmek amacıyla onnx ve TensorRT dönüşümleri yapıldı. Bu süreç, eğitilen modellerin pratik uygulamalarda kullanılabilirliğini artırmak ve gerçek dünya koşullarında başarılı bir şekilde çalışmalarını sağlamak için önemli bir adımdır.

Geliştirilen uygulama, Nvidia Jetson Xavier NX üzerinde başarıyla çalışarak, gerçek zamanlı işleme performansında 12 FPS hızında sorunsuz bir performans sergiliyor. Bu uygulama, otonom araç sistemlerinde kullanılabilecek güçlü bir gömülü cihazın yeteneklerini ortaya koymaktadır. Nesne tespiti ve mesafe tahmini gibi kritik görevleri güvenilir bir şekilde yerine getirebilmesi, 2D tek gözlü kameraların dahi kullanılmasıyla mümkün olduğunu kanıtlamaktadır. Ortalama Mutlak Hata'nın 1.51 metre olması, uygulamanın doğruluğunu ve güvenilirliğini vurgulamaktadır. Bu başarılar, otonom araç teknolojilerinin pratikte kullanımına önemli bir adım sağlar.

1. INTRODUCTION

1.1 Purpose of Thesis

This thesis aims to design a cost-effective, real-time, and reliable system based on deep learning for object detection and distance estimation using 2D monocular cameras in autonomous vehicles. Accurate and fast distance estimation plays a critical role in ensuring safe navigation and enhancing environmental awareness in autonomous driving systems. While stereo or 3D cameras are widely used today, they come with high costs and complex structures. This study aims to provide a more cost-effective alternative by utilizing 2D monocular cameras.

Within this framework, the system utilizes the YOLOv8 (You Only Look Once version 8) model to detect objects and the Monodepth2 model to estimate distances, both optimized to run on the Nvidia Jetson Xavier NX, showcasing real-time performance on embedded systems.Furthermore, a novel Weighted Distance Estimation Algorithm is proposed to reduce the errors caused by the depth differences between the object and the background. This work aims to optimize the use of 2D monocular cameras in autonomous vehicles, providing a cost-effective and high-performance solution.

1.2 Problem

The primary challenge in the development of autonomous vehicles is achieving reliable and accurate object detection and distance estimation in real-time. While various sensing technologies, including stereo and 3D cameras, are available for these tasks, they often involve high costs and complex configurations that may not be practical for widespread deployment in autonomous systems.

Moreover, distance estimation using traditional methods can be hindered by the limitations of 2D monocular cameras, which can lead to inaccuracies due to factors such as varying background depths, occlusions, and the inherent challenges of

interpreting 2D images. The need for precise distance measurements is critical for ensuring safe navigation and effective decision-making in autonomous driving scenarios.

Thus, the problem addressed in this thesis is to develop a deep learning-based system capable of overcoming the limitations of existing methods by using 2D monocular cameras for reliable object detection and accurate distance estimation, while also minimizing the associated costs and complexities of traditional sensor setups.

1.2.1 Research questions

This thesis aims to explore several essential questions concerning the creation and deployment of deep learning models for identifying objects and estimating distances in autonomous vehicles. The following research questions, detailed in Chapter 6, will be examined:

How can deep learning algorithms be effectively utilized for real-time object detection using 2D monocular cameras in autonomous vehicles?

What methodologies can be developed to accurately predict distances to detected objects in monocular camera images while minimizing the impact of background depth variations?

How does the performance of the YOLOv8 model for object detection compare in terms of accuracy and processing speed to other existing models on embedded platforms such as Nvidia Jetson Xavier NX?

What are the advantages and limitations of using the Monodepth2 model for depth estimation in autonomous driving applications when utilizing monocular images?

How can the integration of object detection and distance estimation models enhance the overall reliability and efficiency of autonomous vehicle systems?

1.3 Literature Review

In this section, previous studies on depth estimation are reviewed, highlighting their significance across diverse application domains such as autonomous vehicles. A

variety of innovative approaches, including deep learning techniques and traditional computer vision methods, are employed in these studies, with each contributing unique insights into depth perception. Several methods exist for depth estimation, and they can be broadly categorized into monocular, binocular (stereo), and multi-view techniques. Most of depth estimation approaches need more than one input image or sensors. In this study focuses on monocular depth estimation from single input image.

1.3.1 Monocular approaches

Monocular depth estimation refers to estimating object distances within a scene from just one 2D image. Unlike stereo depth estimation, which relies on two images (stereo pairs) to calculate depth through the disparity between corresponding points, monocular depth estimation attempts to infer 3D information from just one image. This makes the task inherently more challenging because the depth cannot be directly computed without stereo or multi-view information. One particularly interesting study [1] focuses on monocular depth estimation under low-light conditions using thermal images. The authors have developed an image transformation and GAN-based method to predict scene depth using a single thermal image. This method employs deep learning models to learn the relationship between thermal and color images. The results demonstrate the effectiveness of this approach under low-light conditions. Addressing the importance of monocular depth estimation at night and in challenging weather conditions, [2] introduces a new framework called EVEN, which utilizes RGB and depth data. This framework encompasses image enhancement, multi-modal fusion, and depth estimation stages to handle low-light conditions effectively. On the other hand, [3] underscores the difficulty of understanding 3D objects in autonomous vehicles and suggests the use of deep learning in place of expensive sensors like Lidar. Another study, [4], proposes two independent autoencoder-based and detection-based models in parallel. In this work, YOLOv5 is used for finding and classifying objects, while DepthNet predicts depth values from 2D images taken by a monocular camera. DepthNet uses an unsupervised autoencoder network with ResNet-50 as the backbone for depth prediction and ResNet-18 for pose estimation. Graph Convolution Networks are added to the decoder to produce depth images at different scales. The object's

relative distance is calculated by finding the median depth value of all pixels inside the object's bounding box. However, a drawback of this method is that YOLOv5 has difficulty detecting small objects.

1.3.2 Binocular (stereo) approaches

This method uses two or more cameras, typically placed side by side, to capture images from slightly different viewpoints. The disparity (difference in the position of corresponding points in the images) is calculated and used to estimate depth. A novel approach that leverages traditional stereo information for monocular depth estimation from a single image is presented in [5]. The authors have developed a deep learning model called monoResMatch, designed for sharp and dense depth prediction from a single image. The key innovation lies in synthesizing features from a horizontally aligned viewpoint and performing stereo matching between the input image and the synthesized features. Furthermore, traditional stereo algorithms are trained such as Semi-Global Matching (SGM), to generate proxy ground truth annotations, allowing our model to maintain a self-supervised approach while improving depth accuracy. Additionally, the use of proxy labels and training processes are elaborated upon in detail. Another study [6] introduces an approach called STereo TRansformer (STTR). Stereo depth estimation has typically been based on finding the best pixel matches along epipolar lines in the left and right images to estimate depth. In this study, the problem is viewed as a sequence-to-sequence matching task. Instead of the usual cost volume method, dense pixel matching is used with positional data and attention mechanisms. The authors highlight several benefits of STTR, such as removing the need for a fixed disparity range, detecting occluded areas, providing confidence scores, and enforcing uniqueness during matching. Tests on both synthetic and real-world datasets show that STTR works well across different domains, even without fine-tuning. Furthermore, [7] presents a new training approach aiming to surpass existing supervised depth estimation methods. It targets unsupervised depth prediction using epipolar geometry and aims to learn the relationship between left and right images using reconstruction loss and disparity images. One notable work is by [8]. The authors developed two lightweight models for stereo vision that maintain accuracy

while significantly reducing complexity. MobileStereoNets represents end-to-end 2D/3D models based on MobileNet-V1 [9] and MobileNet-V2 [10]. The study demonstrates that the proposed 2D/3D networks significantly lower computational costs, achieving reductions of 27%/95% and 72%/38% in parameters and operations for the 2D and 3D models, respectively, without sacrificing accuracy.

To best of our knowledge, as one of the main differences from the existing studies, our approach utilizes YOLOv8 model for object detection enabling fast response time during the detection and unlocking operation capability in real time. In addition, we use Monodepth2 model hand to hand with YOLO to extract depth map and use specialized algorithm to increase the accuracy of depth estimation. In conclusion, this study aims to demonstrate that deep learning-based monocular depth estimation can cost-effectively and efficiently unlock a great potential for future applications and solutions in the smart mobility environment.

1.4 Contributions

The thesis presents four main contributions. First, we utilize the YOLOv8 model for real-time object detection, validating its performance through comprehensive evaluation. Second, we propose a weighted average approach in depth measurement to eliminate the impact of errors caused by the background within the bounding box during object detection. Third, we achieve more robust depth estimation by employing a Linear Regression model trained with the median values of RGB channels extracted from the weighted depth map. Finally, we integrate the YOLOv8 and Monodepth2 models on the Nvidia Jetson Xavier NX for real-time applications, leveraging optimized GPU performance and the advantages of the TensorRT library. These contributions significantly enhance the existing methods in the fields of object detection and depth estimation.

1.5 Outlines

The outlines of this thesis are structured around the various chapters to provide a comprehensive understanding of the research conducted.

In Chapter 1, the focus is on introducing the topic of object detection and distance estimation, providing a literature review that outlines existing research and methodologies in the field, thereby establishing the significance of the study.

Chapter 2, titled Base Study, presents the foundational theories and technologies, specifically examining the YOLOv8 model for object detection and Monodepth2 for depth estimation, highlighting their relevance and applicability.

Chapter 3, the Methodology, details the processes and techniques employed to integrate these models, outlining the experimental setup and data collection methods used for effective results.

In Chapter 4, the Performance Evaluation, the outcomes of the experiments are analyzed, examining how accurately and efficiently the models perform in different situations.

Chapter 5, Running on IoT Device, discusses the deployment of the developed models on NVIDIA Jetson Xavier NX, illustrating the challenges and solutions encountered in achieving real-time applications in an IoT environment.

Finally, Chapter 6, the Conclusion, synthesizes the findings from the previous chapters, reflecting on the research objectives and suggesting directions for future work in the field of autonomous vehicles.

2. BASE STUDY

2.1 Object Detection

2.1.1 Introduction of object detection

Object detection is a fundamental process of computer vision techniques. This process requires the ***classification of objects present*** in an image and also assigns a corresponding class label to each detected object and draws bounding boxes around them to indicate their positions. Object detection is crucial for a wide range of applications, including but not limited to autonomous driving, video surveillance, robotics, and augmented reality [11].

In the context of this study, object detection plays a pivotal role, particularly in the identification of traffic-related objects for autonomous vehicles. For autonomous vehicles to navigate their environments safely, they must be capable of accurately detecting various objects such as pedestrians, cars, motorcycles, trucks, cyclists, and drivers. Accurate detection of these objects enhances the vehicle's decision-making capabilities, thereby improving road safety and optimizing traffic flow.

Historically, object detection relied on manual feature extraction and statistical learning algorithms. However, these traditional approaches demonstrated limited effectiveness in complex real-world environments, especially when dealing with objects of varying scales, occlusions, and dynamic backgrounds.

The advent of deep learning, specifically CNNs, has marked a significant breakthrough in object detection. CNN-based models automatically learn robust feature representations from raw pixel data, substantially improving both detection accuracy and robustness in diverse scenarios. Modern object detection algorithms are generally categorized into two primary approaches: two-stage detectors and single-stage detectors.

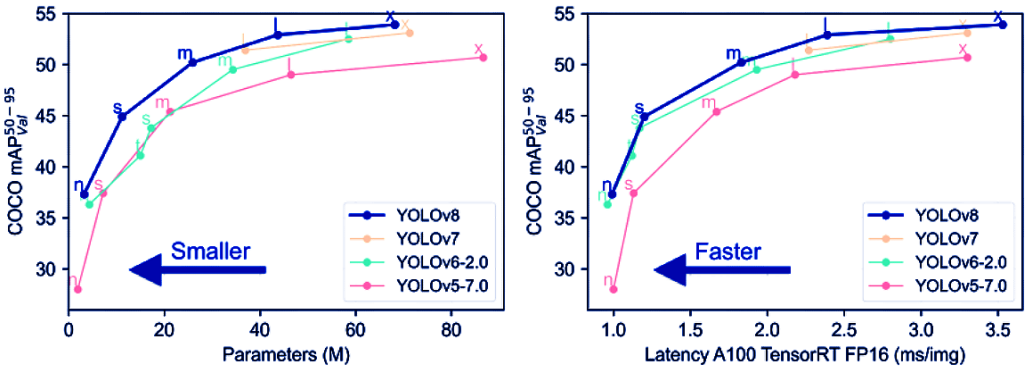

Figure 2.1 : Comparison of YOLO models in terms of performance, parameters, and speed.

Two-stage detectors, such as Faster R-CNN [12], first generate region proposals and then classify each proposed region. In contrast, single-stage detectors, including YOLO and SSD (Single Shot MultiBox Detector) [13], predict bounding boxes and class labels in a single step. Single-stage detectors are particularly advantageous in real-time applications due to their efficiency, and recent advancements have demonstrated improvements in both speed and accuracy.

In this context, the YOLO family stands out for its balance between speed and accuracy. YOLOv8, one of the stable and widely-used versions, has further pushed this balance, becoming one of the leading models in object detection [14].

2.1.2 Comparison of YOLO models

YOLOv8 is one of the most recent versions of the YOLO family, offering significant improvements in speed and accuracy in object detection tasks compared to its predecessors. Thanks to its enhanced architecture and more efficient data processing capabilities, YOLOv8 achieves higher accuracy rates with lower latency. This model not only delivers substantial performance improvements but also stands out due to its reduced computational resource requirements.

Within the model, the available variants—S, M, L, and XL offer different parameter sets and balance between speed and performance, tailored to various use cases. These variants can be optimized based on processor and memory constraints, making the

model suitable for a wider range of applications. In particular, for systems with limited hardware resources, such as IoT devices, the smaller variant, YOLOv8-S, is often preferred for its faster execution with fewer parameters. YOLOv8-S maintains sufficient accuracy while conserving energy in IoT devices due to its ability to operate with lower computational power. In contrast, YOLOv8-XL excels in scenarios that demand higher accuracy, effectively leveraging more powerful hardware to deliver superior performance.

YOLOv8 shows significant advancements in mean average precision (mAP), parameter efficiency, and processing speed when compared to earlier versions, including YOLOv5, YOLOv6, and YOLOv7. These enhancements are visually depicted in 2.1, illustrating the comparative performance across these models and highlighting YOLOv8's superiority in speed, accuracy, and model complexity.

2.1.3 Neural network architectures for YOLOv8 object detection

Designed to deliver high speed and accuracy for real-time object detection, YOLOv8 incorporates architectural improvements and optimization techniques compared to its predecessors. The architecture of YOLOv8, as illustrated in Figure 2.2, comprises four main components: backbone, neck, head, and loss, each contributing to the model's enhanced performance in real-time object detection.

The backbone section utilizes the Cross Stage Partial (CSP) architecture. CSP consists of two parts, with the first part performing convolution calculations while the second part merges the results from the first part. This architecture enhances the learning capabilities of CNNs while reducing the model's computational costs.

YOLOv8 introduces a new C2f module to enrich gradient flow. This module not only provides better learning ability but also reduces computational costs. The model has become more efficient by decreasing the number of blocks at each stage. Additionally, an improved pooling method has been employed to increase inference speed. These changes enhance the overall performance of the model, providing shorter inference times.

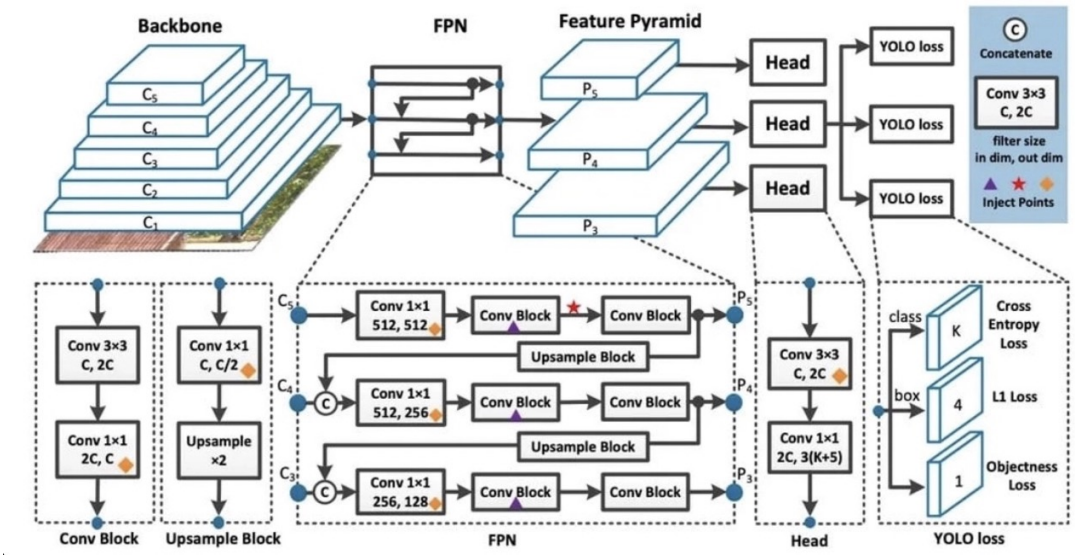

Figure 2.2 : YOLOv8 architecture, consisting of four main components: Backbone, Feature Pyramid Network (FPN), Head, and Loss layers.

In the neck section, as the network depth increases, more feature information is obtained, which improves the quality of dense predictions. However, complicate deep networks may reduce the position information of objects; therefore, Feature Pyramid Network (FPN) and Path Aggregation Network (PAN) architectures gain importance for multi-scale feature fusion. The neck section of the model combines features from different network layers, enriching the upper layers with additional information while preserving positional accuracy in the lower layers through fewer convolutions.

FPN performs upsampling from top to bottom, while PAN downsamples for more information transfer from lower layers to upper layers. The two feature outputs are combined to make accurate predictions for images of various sizes. The model adopts the FP-PAN (Feature Pyramid-Path Aggregation Network) structure, removing convolution operations in upsampling to reduce computational costs.

In the head section, an independent head is used to separate classification and detection tasks compared to the dual-head structure of the YOLOv5 model. This approach allows the model to perform object classification and regression tasks more specifically and accurately. Traditional anchor-based methods use numerous anchors to determine the positions of objects in an image, defining four offsets for each object to adjust their exact locations. However, this study adopts an anchor-free method that defines the

middle of the object and predicts the distance this center to the bounding box. This change improves the computational efficiency of the model while also providing better object detection and classification performance.

The anchor-free method represents a significant distinction from the anchor-based approaches used in previous YOLO versions. While traditional anchor-based methods predict object positions through specific anchor boxes, the anchor-free method eliminates this dependency. In this new approach, the model directly detects the middle of each object and calculates the distance this center to the predicted bounding box. As a result, better performance is achieved in situations where object sizes and aspect ratios vary. This represents a significant innovation and advantage provided by YOLOv8 compared to previous versions.

In the loss section, a Task Aligned Assigner is used to assign positive and negative examples during the model's training. This assigner selects positive examples based on the weighted classification and regression scores. The classification and regression branches in the model aim to enhance performance by utilizing different loss functions. The classification branch employs binary cross-entropy loss, while the regression branch uses losses that expand the value probabilities around the object and consider the aspect ratio of the predicted and ground truth bounding boxes. This structure enables the model to learn more accurately and effectively.

The YOLOv8 model family offers various versions to cater to different size and performance requirements. These versions are categorized by size: nano (n) for minimal resources, small (s) for efficiency, medium (m) for balance, large (l) for high accuracy, and extra large (xl) for maximum performance. Each version strikes a balance to meet specific needs: some applications require faster and lighter models, while others need heavier models with higher accuracy. As a result, smaller models provide faster inference but may exhibit lower accuracy, whereas larger models deliver higher accuracy at the expense of increased computational resource demands.

This distinction arises from the necessity to optimize applications for different hardware environments and requirements. For instance, a lightweight model such as YOLOv8n is preferred for IoT devices with limited computational power, while larger

models like YOLOv8l or YOLOv8xl are better suited for tasks necessitating higher accuracy or more powerful hardware. This variety empowers users to select the model that best fits their specific application.

The mAP results of each version on the COCO [15] val2017 dataset, along with their inference speed measured using ONNX on an Amazon EC2 P4d server, have been previously presented [14]. This analysis demonstrates that as the model size increases, accuracy improves; however, inference time also increases accordingly.

2.2 Depth Estimation

2.2.1 Introduction of depth estimation

Depth estimation is a critical aspect of computer vision that aims to determine the distance of objects from a viewpoint in a given image or scene. This process allows systems to perceive the three-dimensional structure of their surroundings, enabling a more comprehensive understanding of the environment. Depth information is crucial for tasks like autonomous driving, robotics, augmented reality, and 3D reconstruction.

In the context of this study, depth estimation is particularly relevant for autonomous vehicles, where accurately assessing the distance to surrounding objects is crucial for safe navigation. By providing depth information, autonomous systems can better evaluate the spatial relationships between vehicles, pedestrians, and obstacles, leading to enhanced decision-making and improved safety measures.

Historically, depth estimation techniques have relied on stereo vision, which uses at least two cameras to capture images from marginally different viewpoints. Depth information can be found by comparing differences between these images. However, traditional stereo methods often struggle in scenarios with limited texture, occlusions, or significant variations in lighting conditions.

Recent advancements in deep learning have revolutionized depth estimation, with CNNs emerging as powerful tools for predicting depth from single images. These CNN-based approaches leverage large datasets to learn complex depth features, allowing for more accurate predictions in diverse and challenging environments.

Techniques such as monocular depth estimation have gained traction, enabling depth inference using just one camera input, which is particularly advantageous for autonomous vehicles that may have size and weight constraints.

In this context, the Monodepth architecture offers the capability to perform depth estimation using monocular camera images. The Monodepth2 model, developed based on the U-Net architecture [16], is utilized to estimate distances [17]. U-Net is an encoder-decoder network characterized by skip connections, which enhance the learning of depth information. As a result, the accuracy and efficiency of depth estimation are significantly improved.

Monodepth2 adopts a self-supervised learning approach that does not require ground truth depth data. This methodology enables the model to learn complex depth features using information derived from large datasets, allowing for more accurate predictions in various challenging environments. The features of Monodepth2 significantly enhance the applicability of depth estimation in real-world scenarios.

Furthermore, the integration of depth estimation with object detection frameworks enhances the capabilities of autonomous systems. By combining spatial information with object recognition, vehicles can better understand their surroundings and respond more effectively to dynamic situations. This integration plays a critical role in ensuring the safe and efficient operation of autonomous systems.

2.2.2 Neural network architectures for depth estimation

Depth estimation is important for many deep learning applications, including autonomous driving, robotics, and augmented reality. MonoDepth2 represents a significant advancement in this field, employing self-supervised learning techniques to estimate depth by utilizing the geometric relationship between consecutive frames. The model includes a depth network for predicting depth from an image and a pose network for estimating camera motion between frames. These two networks are trained simultaneously using multiple loss functions to minimize the errors in depth and pose predictions.

The architecture of the depth network in MonoDepth2 follows a U-Net-like structure. The encoder is based on a pre-trained ResNet [18] model, which extracts meaningful features from the input images. The decoder then processes these features to generate depth maps, converting the normalized sigmoid output into metric depth values using a logarithmic scaling function. The pose network, on the other hand, estimates the camera's relative pose between consecutive frames, enabling the system to perform self-supervised learning by utilizing the photometric consistency between frames.

The training process relies on learning the disparity between the outputs of the depth and pose networks. One of the key loss functions used is the photometric reconstruction loss [19], which evaluates the difference between the original input images and the synthesized images reconstructed from the predicted depth and pose. This loss is calculated across multiple scales using image pyramids, enhancing the model's ability to capture both global and local features more effectively.

In summary, the technique, which combines a U-Net-based depth network and a ResNet encoder with advanced multi-scale training methods, allows for accurate depth estimation from monocular images without requiring ground truth depth data during training.

In the Monodepth2 project, the U-Net architecture is employed for depth estimation, which is a popular choice for handling spatial information in image processing tasks. The architecture consists of an encoder-decoder structure that efficiently preserves fine details while estimating depth from monocular images.

The encoder part of U-Net applies successive convolutional layers combined with max-pooling operations to extract hierarchical features from the input image. Each pooling layer reduces the spatial resolution while increasing the number of filters, allowing the network to learn more abstract features. In the context of Monodepth2, this enables the model to capture essential elements such as objects and road structures, which are critical for autonomous vehicles.

The decoder reverses the process by using up-sampling layers to restore the spatial resolution, assisted by skip connections that bring back low-level features from the encoder. These connections are essential for maintaining sharp boundaries and fine

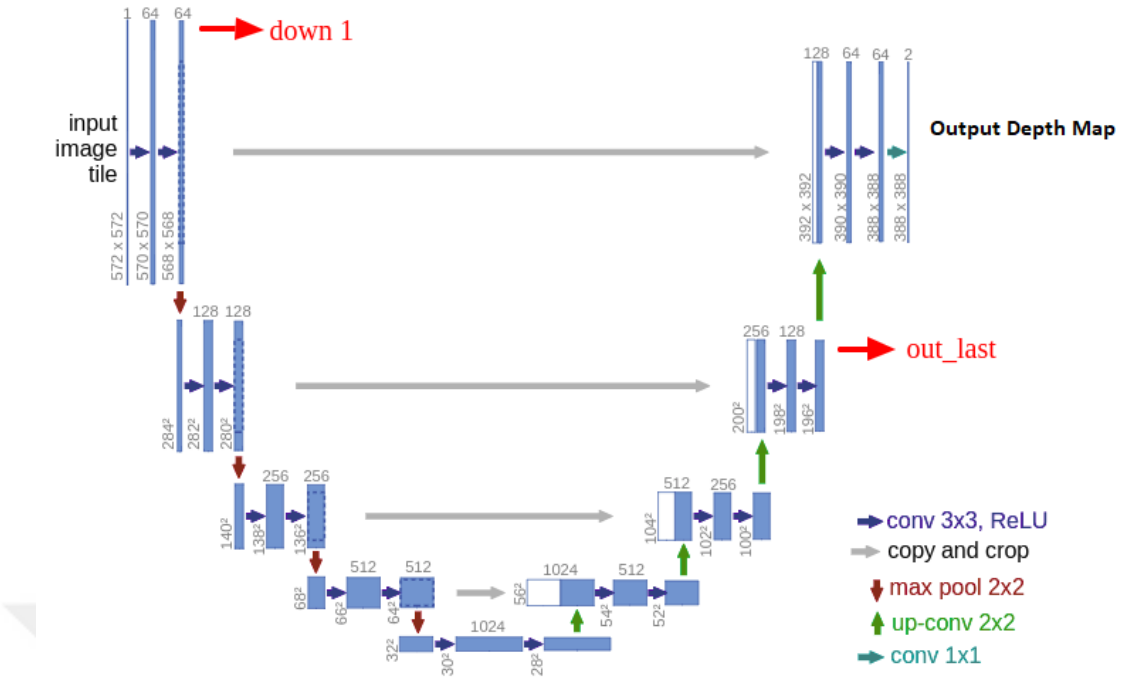

Figure 2.3 : This figure illustrates the U-Net architecture. The U-Net model downsamples and then upsamples the input image to generate precise, pixel-level predictions.

details in the predicted depth maps, which are crucial for applications like autonomous driving where accurate distance measurements are needed.

U-Net's symmetric design ensures that minimal spatial information is lost, which is particularly beneficial in depth estimation tasks that require high-resolution outputs. By maintaining spatial consistency through skip connections, the model improves its capacity to predict precise depth, leading to better performance in tasks like object detection and distance calculation in autonomous vehicles. This structure is depicted in the U-Net architecture diagram shown in 2.3.

PoseNet is a crucial architecture used in the Monodepth2 project for predicting camera motion. This architecture estimates the position and rotation in the context of six Degrees of Freedom (DoF) between two consecutive color image frames [20]. PoseNet is fundamentally based on the ResNet18 architecture and takes two images from consecutive time steps, specifically the frames at $t-1$ and t, as input [21].

PoseNet is a critical component for relative pose estimation between two consecutive frames. Designed to model the camera's motion between these two color images, PoseNet utilizes temporally consecutive image pairs to determine the relationship between the current frame (t) and the previous frame ($t-1$), unlike typical stereo image pairs. This process represents the camera's position in the six DoF framework and employs an axis-angle representation for pose estimation.

In the Monodepth2 architecture, the 6-DoF pose estimation plays a vital role in predicting the relative motion between consecutive frames. The 6-DoF model represents the camera's movement within the scene across three translational axes (x, y, z) and three rotational axes (roll, pitch, yaw) [22]. This comprehensive representation enables the model to understand changes in position and orientation within the scene and is essential for depth estimation from monocular sequences. PoseNet generates a single 6-DoF relative pose information when predicting the rotation between two frames, which includes both rotational and translational components.

In Monodepth2, the pose estimation network is constructed using a ResNet18 architecture that takes two consecutive monocular images as input. These sequential frames assist the model in predicting the relative pose between the current frame and the previous frame. This prediction between consecutive images contributes to the overall learning process by integrating with the depth predicted by the Depth Network to compute losses. PoseNet outputs rotation and translation vectors that define the camera's motion.

The ResNet18 architecture is a CNNs widely used in deep learning applications and is optimized for depth learning. One of the key features of the ResNet architecture is its use of "skip connections" that directly link layers. This structure allows for more effective learning by reducing the "gradient vanishing" problem that arises during the training of deeper networks. Offering an 18-layer deep structure, ResNet18 is optimized for extracting features from 2D images. Each convolutional layer learns local features from the image, while the accumulation of these features is combined at deeper layers to reach high-level abstractions.

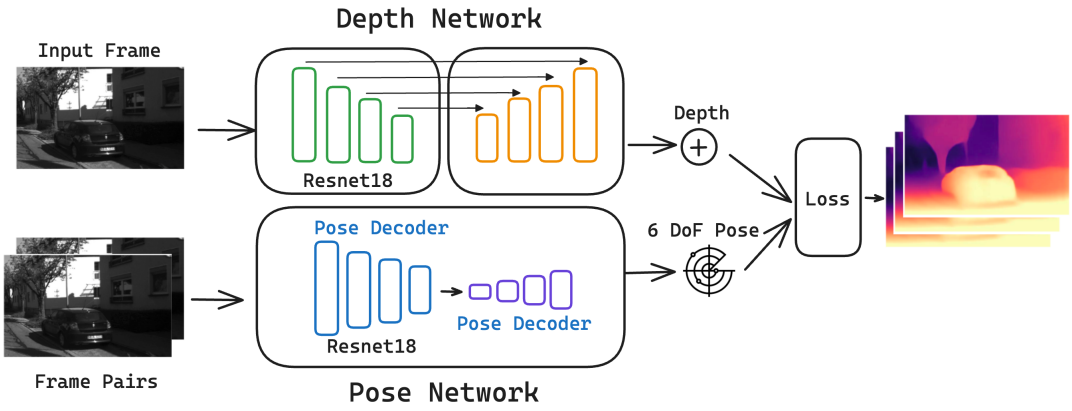

Figure 2.4 : The Monodepth2 architecture includes a depth network and a pose network. The depth network uses a ResNet18-based encoder-decoder to predict depth from a single image. The pose network takes consecutive frame pairs and predicts the 6-DoF camera pose. Both outputs are used to compute a self-supervised loss for depth estimation.

This study presents an architecture designed for performing depth estimation from monocular images. The architecture aims to predict both the depth map of a scene and the camera motion using a sequence of images captured by a single camera. It consists of three main components: the Depth Network, the Pose Network, and the Loss Function.

The Depth Network is responsible for estimating depth from a single image frame. This network is based on a ResNet18, which extracts multi-layer feature maps from the input image. These features are then processed in the decoding stage to generate a depth map. As a result, each pixel in the input image is assigned a depth value, allowing for the estimation of the distance of each object in the scene from the camera.

The Pose Network estimates the relative pose difference (6-DoF) between two consecutive image frames. Also based on ResNet18, this network learns the rotational and translational movements between the frames. The extracted features from the input frames are processed by the pose decoder to predict the relative pose between the two images.

The Loss Function, referred to as the Photometric Reconstruction Loss, combines the depth and pose estimates. It minimizes the photometric difference between the two images, thereby improving both depth and pose predictions. By optimizing through

this loss function, the Depth Network and Pose Network work collaboratively to predict both the depth map of the scene and the camera motion. Consequently, depth estimation can be performed using only monocular images. The overall structure and data flow of this architecture are illustrated in Figure 2.4.

2.3 Open Neural Network Exchange

2.3.1 Overview of ONNX

Open Neural Network Exchange (ONNX) is an open-source intermediary format that facilitates the transfer and execution of various machine learning and deep learning models across different frameworks [23]. ONNX enables the universal export of models developed on diverse platforms, such as PyTorch, TensorFlow, and MXNet, allowing for their deployment in varied environments. This feature provides flexibility not only during the model development phase but also throughout the deployment and usage processes. With its capability to perform efficient inference and optimization, ONNX enhances performance across various hardware environments (GPU, CPU, IoT devices) [23].

2.3.2 Utilization of ONNX models in IoT devices

The utilization of ONNX format models in IoT devices presents significant advantages regarding memory and processing power, particularly given the limited resources of these devices. ONNX format models achieve high performance when deployed in IoT devices due to their lightweight and rapid processing capabilities. Consequently, ONNX models can be optimized through platforms like TensorRT or ONNX Runtime, enhancing their efficiency for IoT devices characterized by low power consumption and limited computational capacity. TensorRT is a library developed by NVIDIA designed to enhance the performance of deep learning models. This library enables model optimization and facilitates rapid inference capabilities, allowing devices to utilize their limited resources efficiently.

Another essential optimization method is quantization, which involves representing the parameters and computations of machine learning and deep learning models using lower bit widths. For instance, employing 8-bit integers instead of 32-bit floating-point numbers can reduce the model's memory footprint while improving computation time and energy consumption. This method offers substantial advantages, particularly for

low-power devices and IoT applications, as it enhances capabilities for faster and more efficient processing in resource-constrained systems. Quantization is achieved by quantizing both weights and activations, thereby improving performance in deep learning applications.

Thanks to these optimizations, rapid and effective inference can be performed in tasks such as deep learning-based image processing, speech recognition, and anomaly detection. Particularly in edge computing environments, the low latency and efficient inference capabilities provided by ONNX support real-time decision-making processes in IoT devices.

3. METHODOLOGY

3.1 Data Sets and Preprocessing

This section introduces the datasets utilized in this study for training and evaluating the object detection and distance estimation models. The BDD100K [24] and KITTI [25] datasets are widely recognized in autonomous driving research and provide diverse data that enhances the robustness of deep learning models. Additionally, we will describe the preprocessing techniques applied to these datasets to prepare them for use in the models.

3.1.1 BDD100K datasets

BDD100K is an important dataset for autonomous driving and computer vision research. This dataset consists of over 100,000 videos and 1.5 million frames, featuring RGB images with a resolution of 1280x720, covering various scenarios such as different weather conditions, time periods, and locations. Figure 3.1 shows sample images from the BDD100K dataset. BDD100K contains labeled image and data for tasks like object detection, lane detection, and semantic segmentation.

Figure 3.1 : Samples from the BDD100K dataset.

The dataset includes diverse traffic scenarios in both urban and rural areas, providing important information such as various vehicle types, pedestrians, cyclists, as well as

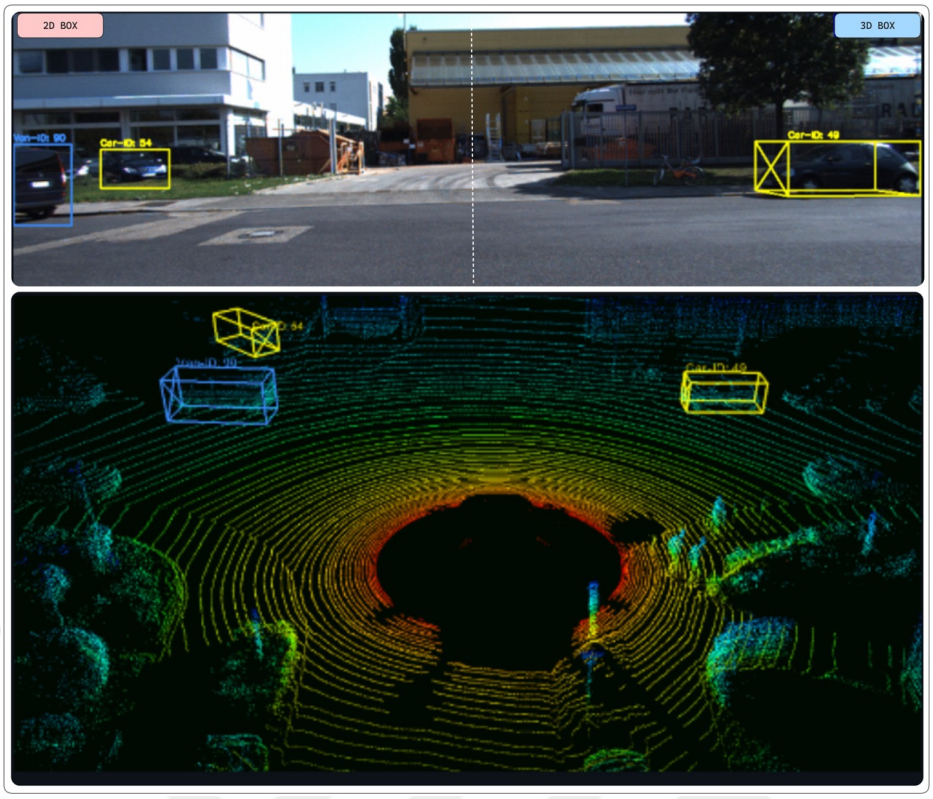

Figure 3.2 : In the KITTI dataset, objects are annotated with 2D and 3D boxes, along with corresponding 3D Velodyne LiDAR points.

road signs and lane markings. This extensive range of data helps deep learning models become more robust and generalizable, offering the diversity needed for autonomous driving systems to operate safely and effectively.

3.1.2 KITTI datasets

The KITTI dataset is a significant resource for autonomous driving research, providing images captured by stereo cameras mounted on a vehicle. This dataset is particularly valuable for depth estimation tasks, as it includes RGB images obtained from a stereo camera system and depth maps derived from Velodyne LiDAR. Additionally, it offers labeled data for tasks such as object detection, visual odometry, 3D object detection, and scene flow estimation. The object detection labels include cars, vans, trucks, pedestrians, sitters, cyclists, trams, and miscellaneous objects. The visualization of 2D and 3D object detection annotations, along with LiDAR data, is shown in 3.2.

The stereo images captured from left and right perspectives form the basis for depth estimation, while the depth maps derived from Velodyne LiDAR point clouds enable

accurate depth prediction for scenes and objects. The 3D Velodyne point clouds, which contain between 90,000 and 100,000 points per frame, provide critical spatial information for 3D object detection and environmental mapping. Moreover, the OXTS GPS/IMU data supplies the vehicle's position, speed, orientation, and acceleration, supporting visual odometry and trajectory prediction tasks. The video categories include city, residential, road, campus, person, and calibration. Labeled objects, such as vehicles, pedestrians, cyclists, and trucks, are provided in both 2D and 3D formats, enhancing object detection and tracking. Calibration files ensure the alignment of RGB images, depth maps, and point clouds, allowing for accurate mapping of 2D detections to 3D space.

3.1.3 Augmentation and preprocessing methods

The preprocessing stages of the BDD100K dataset begin with parsing the image annotations from the JSON format [26], where labels, categories, and bounding box coordinates associated with each image are processed. The images are then resized and normalized to meet the model's requirements. Data augmentation techniques such as rotation, shifting, and flipping are applied to ensure the model is trained on a more diverse dataset. For the autonomous vehicle model, specific labels including person, rider, car, bus, truck, bike, and motorcycle were used to enhance road and traffic safety, while other labels were excluded from the process. These selected labels are critical in recognizing dynamic objects around the vehicle, ensuring safe driving and accurate navigation. Finally, the dataset is split into training and testing sets, making it suitable for model training and evaluation.

The Eigen method, as described by Eigen et al. [27], was employed to divide the KITTI dataset into training, validation, and test sets, consisting of 39,810, 4,424, and 697 images, respectively. This ensured a balanced dataset for accurate model evaluation. During the preprocessing stages, images were converted from PNG to JPEG format, maintaining image quality at 92% while optimizing file size using a sampling factor of 2x2, 1x1, 1x1. The raw depth maps from the Velodyne LiDAR sensor were resized, and both the images and depth maps underwent horizontal flipping for data augmentation. The input resolution of the images was set to 1024x320 pixels

to enhance computational efficiency and model accuracy. Additionally, static frames were removed, retaining only dynamic frames containing object and scene changes, thereby improving the model's generalization capability.

3.2 Object Detection

YOLOv8 is an optimized version of the YOLO object detection model, designed for high-speed and high-accuracy detection across a range of object scales. YOLOv8 introduces several architectural improvements over its predecessors, such as the use of CSPDarknet53 [28] as the backbone and the Bi-FPN [29] structure for efficient feature fusion. These innovations enable the model to effectively detect objects in complex environments while maintaining low computational overhead.

YOLOv8 takes features from input images, uses them to draw boxes around objects, and identifies their classes. It employs the CSPDarknet53 [28] network as its backbone, offering effective learning capabilities. The model also incorporates a PAN-FPN [30] structure, known as the bidirectional feature pyramid network (Bi-FPN), at the neck, which facilitates efficient multiscale feature fusion. The Bi-FPN structure uses adjustable weights to combine features from top to bottom and bottom to top, helping the network focus on key features at different scales. Furthermore, YOLOv8 uses a scaling method that evenly adjusts the resolution, depth, and width of all network parts, ensuring high performance even with limited resources.

The loss function in YOLOv8 consists of three main components: the classification loss (L_{cls}), the bounding box regression loss (L_{box}), and the distribution focal loss (L_{dfl}). These components are combined into a final loss function, expressed as the weighted sum of the individual losses:

$$L_{\text{total}} = L_{\text{box}} + L_{\text{dfl}} + L_{\text{cls}} \tag{3.1}$$

The weights of these components are hyperparameters that are adjusted during training to balance the importance of each loss term. For instance, the classification loss weight (L_{cls}) controls the impact of class prediction accuracy, while the bounding box regression loss (L_{box}) influences the precision of the predicted bounding boxes. The

distribution focal loss (L_{dfl}) helps address class imbalance and enhances bounding box regression, particularly for objects with blurry or unclear boundaries.

3.2.1 Binary cross entropy

Binary Cross Entropy (BCE) is a widely used loss function for binary classification tasks, such as object detection, image segmentation, and other tasks that involve two classes. It quantifies the difference between the predicted probabilities and the true binary labels. BCE is particularly effective when the model outputs a probability score, as it offers a clear and intuitive measure of how well the model's predictions align with the actual labels [31].

The Binary Cross Entropy loss function is given by:

$$\text{BCE}(p, y) = -\left(y \log(p) + (1-y) \log(1-p)\right) \tag{3.2}$$

In this equation, p represents the predicted probability of the positive class, and y denotes the true label, where $y \in \{0, 1\}$. The function calculates the log-likelihood of the predicted probabilities for the true binary labels. Specifically, when the true label is $y = 1$, the loss is determined by $\log(p)$, and when $y = 0$, the loss is calculated using $\log(1-p)$.

The BCE loss penalizes predictions that deviate from the true labels, with larger penalties for more significant deviations. This property encourages the model to output probabilities that are as close as possible to the actual labels. As a result, BCE is an effective loss function in scenarios where a clear binary distinction is needed, such as detecting the presence or absence of objects in object detection tasks.

In practice, BCE is frequently paired with the sigmoid activation function, which constrains the output probabilities to the [0, 1] range. The combination of BCE and the sigmoid function provides a robust framework for training binary classification models, ensuring that the model's outputs are interpretable as probabilities [31].

3.2.2 Complete intersection over union

Complete Intersection over Union (CIoU) is a metric designed to enhance the accuracy of bounding box predictions in object detection tasks. Unlike the traditional Intersection over Union (IoU), CIoU takes into account not only the overlap ratio but also the center distance between predicted and ground truth boxes, as well as the alignment of their aspect ratios. This multi-faceted approach allows for more accurate localization of the bounding boxes. CIoU is commonly employed in state-of-the-art object detection algorithms, such as YOLOv5 and YOLOv8, as it helps to improve the precision of object detection by ensuring that bounding boxes are better positioned and aligned.

The CIoU loss function is expressed as:

$$\text{CIoU} = 1 - \text{IoU} + \frac{\rho^2(b, b_g)}{c^2} + \alpha v \tag{3.3}$$

In this formula, IoU represents the overlap ratio between the predicted bounding box b and the ground truth bounding box b_g. The term $\rho(b, b_g)$ denotes the Euclidean distance between the center points of b and b_g. The variable c is the diagonal length of the smallest box that contains b and b_g. The term v checks the aspect ratio consistency between b and b_g, and α sets the weight for the aspect ratio term.

The primary advantage of CIoU is that it integrates multiple factors—overlap, center distance, and aspect ratio alignment—into a single loss function. This enables the model to not only classify objects correctly but also to predict bounding boxes that are well-positioned and accurately sized. The inclusion of center distance and aspect ratio alignment ensures that predicted boxes are closer to the target boxes, facilitating faster and more accurate learning during training. Additionally, the CIoU metric optimizes the shape and size consistency of the bounding boxes, leading to more precise object localization results [32].

3.2.3 Distribution focal loss

Distribution Focal Loss (DFL) is a loss function developed to improve the precision of bounding box coordinates in object detection [33]. Instead of directly predicting the bounding box coordinates, DFL enhances accuracy by making predictions based on probability distributions. This approach allows the model to perform distribution-based regression for each coordinate, ensuring that the predicted bounding boxes are closer to the true positions. DFL is commonly used in models such as YOLOv8 to improve bounding box accuracy.

DFL computes a weighted focal loss based on the distance between the target position and the two adjacent bins closest to it. The mathematical expression for the distribution-based focal loss is as follows [33]:

$$\text{DFL}(S_i, S_{i+1}) = -(y_{i+1} - y)\log(S_i) + (y - y_i)\log(S_{i+1}) \tag{3.4}$$

In this formula, S_i represents the probability predicted by the model for the ith bin, and S_{i+1} is the probability predicted for the $(i+1)$th bin. The true bounding box coordinate y is compared with the nearest bin values y_i and y_{i+1}. The distances $y_{i+1} - y$ and $y - y_i$ between the true value and the predicted bins are considered and are weighted accordingly in the loss function. Thus, the loss is minimized based on the proximity to the target bin.

The bounding box coordinates are divided into discrete bins, allowing the model to learn which bin a coordinate is closest to, rather than predicting the coordinate as a continuous value [33]. The model then makes probability predictions for each bin, which are normalized using the softmax function. DFL is computed based on the probability distribution between the two bins closest to the true coordinate. In the formula, the logarithmic probabilities of the bins closest to the true value are weighted by their respective distances and summed. This approach optimizes the bins closest to the target and penalizes incorrect bin predictions. As a result, the model not only learns to predict the correct class but also optimizes the likelihood of each bounding

box coordinate being in the correct position. This method helps the model predict object boundaries with greater precision.

This technique ensures that the model learns not only the correct class predictions but also optimizes the likelihood of each bounding box coordinate being in the correct location. Consequently, the model learns the probabilities of all regions near the target coordinate, rather than just predicting an exact value. This allows the model to make more accurate predictions for object boundaries.

3.3 Depth Estimation

3.3.1 Self-Supervised depth estimation

Self-supervised depth estimation is a significant approach in computer vision, particularly in scenarios where ground truth depth data is not available. In this section, we explore the fundamental principles of self-supervised depth estimation, outlining how the task is framed as a problem of novel view synthesis. For detailed information on the architecture and components of the network used, please refer to Section 2.

Self-supervised depth estimation treats the learning task as novel view synthesis. This method trains a network to predict how a target image would appear from the perspective of another image. By limiting the network to synthesize images using intermediate variables like depth or disparity, it becomes possible to derive meaningful depth information from the model. However, this is an ill-posed problem because many incorrect depth values per pixel can still reconstruct the image accurately. Due to the relative pose between two views, there is significant uncertainty in precisely reconstructing the novel view.

During training, we define the problem based on the minimization of photometric reprojection error L_p. We define the relative pose between the target image I_t and the source image I_{t_0} as $T_{t\to t_0}$. Using the depth map D_t, we aim to minimize the L_p value.

3.3.2 Minimum reprojection loss

In existing self-supervised depth estimation methods [34,35], reprojection error from multiple source images is typically computed and then averaged for each source image. However, some pixels visible in the target image may not be visible in certain source images. In such cases, even if the pixel has an accurate depth prediction in the target image, its color in the occluded source image will differ from the target, causing a high photometric error penalty. These problematic pixels often arise due to out-of-view areas at image boundaries, caused by egomotion, or due to occlusion.

The proposed minimum reprojection loss approach addresses these issues by selecting only the minimum error value across all source images for each pixel, rather than averaging the photometric error. This method simultaneously addresses problems with both out-of-view pixels and occlusion-related inconsistencies. The per-pixel photometric loss is computed as follows:

$$L_p = \min_{t_0} p_e(I_t, I_{t_0 \to t}). \tag{3.5}$$

This minimum reprojection loss reduces artifacts at image boundaries, sharpens occlusion edges, and improves overall accuracy. By decreasing blurriness at image boundaries, this method provides sharper depth discontinuities and enhances accuracy.

3.3.3 Auto masking

Self-supervised monocular training typically assumes a moving camera and a static scene. When these situations are broken, such as with a stationary camera or moving objects, the performance may degrade. During training, this can lead to "infinite depth holes" in the predicted depth maps for moving objects.

The automatic masking method filters out pixels that do not undergo appearance changes between consecutive frames, preventing the network from considering objects that move at the same speed as the camera. Additionally, it allows for the disregard of all frames in monocular videos when the camera is stationary.

In this method, a mask μ is applied to the loss function for each pixel. This mask is binary, computed automatically during the network's forward pass, and is not learned from object motion. Pixels that remain unchanged between consecutive frames typically represent a static camera, an object moving at the same speed as the camera, or a low-texture region. Consequently, the mask is adjusted only for pixels where the reprojection error of the warped image $I_{t_0 \to t}$ is lower than that of the original source image I_{t_0}. Mathematically, this situation can be expressed as follows:

$$\mu = \begin{cases} 1 & \text{if } \min_{t_0} p_e(I_t, I_{t_0 \to t}) < \min_{t_0} p_e(I_t, I_{t_0}) \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

When the camera and another object move at a close speed value, the μ mask avoids the loss calculations for static pixels in the image from being affected. When the camera is stationary, the mask is capable of filtering out all pixels. This simple and low-cost adjustment to the loss function leads to significant improvements in performance.

3.3.4 Network architecture and configuration

The depth estimation network utilized in this study is based on a general U-Net [36] architecture with an encoder-decoder structure that includes skip connections, enabling the representation of both deep abstract features and local information. For the encoder, the ResNet18 model was chosen, which contains approximately 11 million parameters. This model offers a faster and more lightweight alternative to larger and slower models, such as DispNet and ResNet50 [19], often used in existing studies. Instead of training from scratch, the model was trained using weights pre-trained on the ImageNet dataset [37], which contributed to improved accuracy.

The depth decoder is configured similarly to models in the current literature, with a sigmoid activation function at the output layer and ELU activation functions elsewhere. Sigmoid outputs are converted to depth values using

$$D = \frac{1}{a\sigma + b}, \tag{3.7}$$

where a and b are chosen to constrain the depth values between 0.1 and 100 units. Additionally, reflection padding, rather than zero padding, is employed in cases where samples fall outside image boundaries, reducing edge artifacts observed in prior works.

For pose estimation, an axis-angle representation is used for rotation prediction, with rotation and translation outputs scaled by a factor of 0.01. During monocular training, a sequence of three frames is utilized, and the pose estimation network is modified to accept two color images (six channels) as input, enabling it to predict a relative pose with a single six DoF.

3.4 Weighted Distance Estimator Algorithm

Within the bounding boxes created by object detection, background imagery often extends beyond the detected object's boundaries. Background pixels may potentially distort the average depth estimation. To reduce the background pixels impact on depth estimation, a Weighted Distance Estimator Algorithm is described as equations (3.8), (3.9), (3.10). The Weighted Distance Estimator Algorithm is focused center image of the detected bounding box, as shown in Figure 3.3.

$$LW = Weight_{\alpha} * (A(BoundingBox) - A(ShrinkBox)) \tag{3.8}$$

$$HW = Weight_{\beta} * A(ShrinkBox) \tag{3.9}$$

$$Depth = LW + HW \tag{3.10}$$

To focus on the object at the center of the bounding box, the bounding box has been cropped according to a reduction factor, and this region is referred to as the *ShrinkBox*. *BoundingBox* represents with yellow color and *ShrinkBox* represents with blue color as shown in Figure 3.3. Using parameters α and β, the outer edges of the detected object are weighted by the α, while the center points are weighted by the β. In these formulas, LW (Low Weight) represents the area difference weighted by α, which is calculated by subtracting the area of the *ShrinkBox*, $A(ShrinkBox)$, from the area of the *BoundingBox*, $A(BoundingBox)$. HW (High Weight) represents the area of the center box weighted by β and is calculated as the area of the *ShrinkBox*. The Depth value represents the sum of LW and HW values, reflecting the combined effect of low and high-weighted pixels. This approach helps to illustrate the influence of pixels with different weights. The reduction factor, $Weight_{\alpha}$ and $Weight_{\beta}$ have been determined experimentally and is explained in the experimental analysis section.

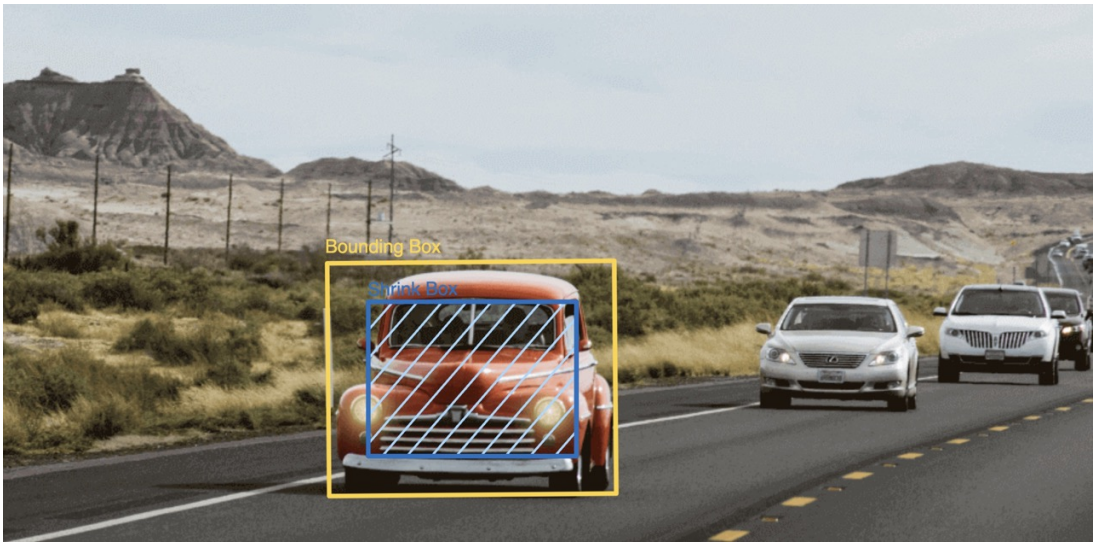

Figure 3.3 : Example for bounding box and shrink box

3.5 Distance Estimation From Depth Matrices

Objects were detected, and their positions were determined using the YOLOv8 algorithm. Subsequently, the Monodepth2 algorithm was applied to estimate the depths of the objects. To minimize background noise, we computed the weighted average of depth matrices. Further refinement was achieved by developing a Linear Regression model, which utilized these depth matrices to make accurate relative distance predictions.

Depth matrices represent depth using color codes. These color codes are used as features for the linear regression model, while distance values are used as targets to build the model. To train the Linear Regression model, depth maps of 1500 images were extracted from the KITTI object detection dataset. The depth maps were resized to match the original image dimensions and normalized to values between 0 and 255. Median values for each color channel were then calculated. The Linear Regression model was trained using these median values and absolute distance data to establish a relationship between relative and absolute distances. The Linear Regression coefficient formula is given in Equation (3.11).

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3 \tag{3.11}$$

In equation (3.11), where the medians of the red, green, and blue channels are represented by the variables x_1, x_2, and x_3, respectively. The values of a_0, a_1, a_2 and a_3 represent the coefficients in the formula. The values are calculated as follows: $a_0 = 58.639, a_1 = -0.137, a_2 = 0.073$ and $a_3 = -0.219$. Thus, this model allows us to make relative distance predictions independently of the object's type, shape, size, and focal length.

The linear regression model determines these coefficients using the least squares method. This method aims to minimize the sum of squared errors, seeking to find the most suitable coefficients by minimizing the squared sum of errors. Based on this idea, curve fitting and least-squares optimization are used to find the approximate values of the four unknown coefficients. This solution is the best way to link absolute distance and relative distance.

4. PERFORMANCE EVALUATION

4.1 Experimental Setup

In this study, tasks related to estimating depth and detecting objects were carried out using the PyTorch library. For object detection, the YOLOv8 model was employed to detect seven distinct object classes. Model training was conducted with a learning rate of 10^{-5}, using the ADAM optimization algorithm and a batch size of 32. To prevent overfitting, early stopping was applied with a maximum of 300 epochs and a patience value of 15. To balance computational efficiency and performance, the lightweight variant of YOLOv8 (YOLOv8s) was selected.

Additionally, the Monodepth2 framework was used for depth estimation, also implemented using PyTorch. Both models were trained on A100 PCIe GPUs with 30 GB of memory, where the Monodepth model required 32 hours and the YOLOv8 model 12 hours to complete training. The training process was monitored using TensorBoard, providing real-time visualization of loss functions and performance metrics.

This experimental setup was optimized for both computational efficiency and accuracy, specifically targeting real-time applications such as object detection and depth estimation on resource-constrained IoT platforms.

A key component of the experimental setup was the deployment of the trained models on an IoT device optimized for real-time applications. In this study, the object detection and depth estimation models were deployed on the NVIDIA Jetson Xavier NX, a device known for its low power consumption and compact design, yet offering significant computational power. To further enhance model performance and efficiency, the TensorRT optimization library was utilized. TensorRT accelerated the inference processes, ensuring faster and more efficient operations, particularly in resource-limited IoT environments.

The software and library versions used in this experiment were as follows: Python 3.10.12, YOLO (ultralytics==8.2.16), TensorRT==10.0.1, PyTorch==2.0.1, torchvision==0.15.2, and TensorBoard==2.14.0.

4.2 Evaluation Metrics

In this study, various metrics are employed to evaluate the performance of object detection and depth estimation models. These metrics are essential for measuring the accuracy, precision, and overall success of the models. The descriptions and formulas for each metric used are provided below.

4.2.1 Intersection over union

IoU is a metric that measures how much the detected object's bounding box overlaps with the ground truth bounding box. It is commonly used to evaluate the accuracy of object detection models by calculating the ratio of the intersection area to the union area of the two boxes. When the IoU value is close to 1, the model is considered to have made an accurate detection; on the other hand, a value close to 0 suggests an incorrect detection.

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{4.1}$$

where $|A \cap B|$ represents the intersection area of the detected and ground truth bounding boxes, and $|A \cup B|$ denotes their union area.

4.2.2 Mean average precision

Mean Average Precision (mAP) is used to measure how accurate a model is in object detection. Specifically, mAP@0.5 checks the model's performance when the IoU threshold is set to 0.5. The IoU value indicates how much the area of detected bounding box intercepts with the area of ground truth bounding box. This metric calculates the ratio of true positives to false positives, then computes the average precision.

$$\text{mAP@0.5} = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{4.2}$$

where N is the number of object classes, and AP_i is the Average Precision for each class.

4.2.3 Precision

Precision is the ratio of true positive predictions to the total positive predictions made by the model. This metric indicates how well the model avoids false positives.

$$\text{Precision} = \frac{TP}{FP + TP} \tag{4.3}$$

where FP is the number of false positive predictions, and TP is the number of true positive predictions.

4.2.4 Recall

Recall is the ratio of true positive predictions to the total number of actual positive instances. This metric measures the model's success in detecting all relevant objects.

$$\text{Recall} = \frac{TP}{FN + TP} \tag{4.4}$$

where FN is the number of false negatives predictions, and TP represents the number of true positive predictions.

4.2.5 Root mean squared error

Root Mean Squared Error (RMSE) evaluates the deviation of the predicted depth values from the actual depth values. It is calculated by taking the square root of the mean of the squared errors, providing a measure of the model's prediction accuracy. A lower RMSE indicates better predictive performance.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{4.5}$$

where N denotes the total number of samples, y_i is the actual depth value, and $\hat{y}_i$ is the predicted depth value.

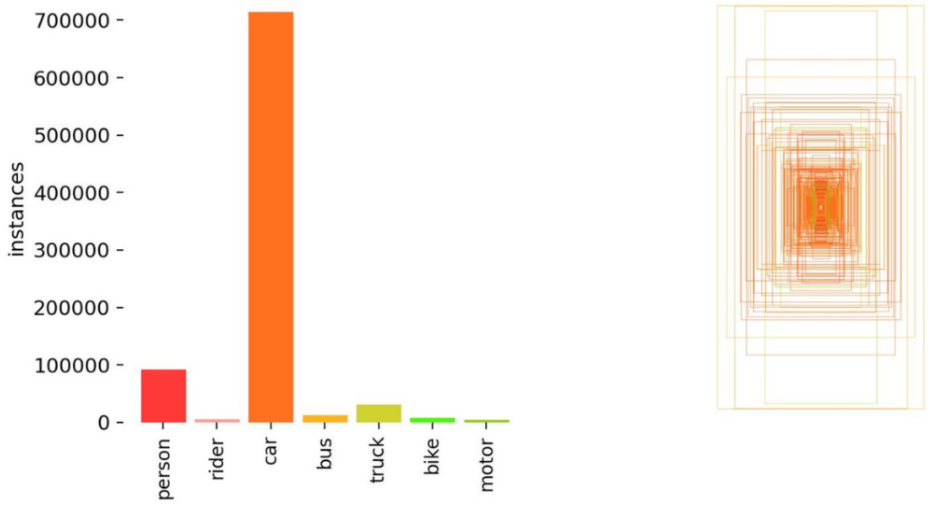

Figure 4.1 : Class distribution and bounding box density analysis for the YOLOv8s object detection model's training dataset.

4.2.6 Absolute error

Absolute Error calculates the average of the absolute differences between the predicted and actual depth values. This metric helps evaluate the overall error rate of the model.

$$\text{Abs} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{4.6}$$

where N is the number of samples, y_i is the actual depth value, and $\hat{y}_i$ is the predicted depth value.

4.3 Experimental Results

In this section, performance metrics, dataset analyses, and evaluation results are presented for the object detection and depth estimation models. The experimental findings are supported by visual data, including class distribution, bounding box density analyses, training accuracy, and loss curves. These results highlight the accuracy of the models in object detection and distance estimation, demonstrating their contributions to real-time autonomous systems.

The object class distribution and bounding box density of the YOLOv8s object detection model's training dataset are shown in Figure 4.1. The bar graph in this

figure reveals an imbalance across classes: the "car" class, with over 700,000 instances, dominates the dataset, while other classes are underrepresented. This imbalance reflects the dataset's focus on road environments, where vehicles are the most common objects. Consequently, the model achieves higher accuracy for the "car" class, whereas detection performance is relatively lower for less-represented classes.

The bounding box density graph in Figure 4.1 indicates that object detections are primarily concentrated near the center of the image. This trend can be attributed to the central focus of in-vehicle cameras, which are designed to capture road and surrounding objects. However, detection accuracy for objects located near the edges of the image could be improved by incorporating more diverse training data, particularly for edge cases and underrepresented classes.

Figure 4.2 presents the spatial and size distributions of bounding boxes in the YOLOv8s model's training dataset. The first plot illustrates the density of object locations, with the y-axis representing vertical positioning and the x-axis representing horizontal positioning. The concentration of objects near the center of the image indicates that the model is primarily trained on centrally located objects. This central focus arises from in-vehicle cameras' tendency to capture road and surrounding objects in the center, which enhances the model's capability to detect objects directly in front of the vehicle. The second drawing shows the distribution of bounding box dimensions in terms of width (x-axis) and height (y-axis). A dominance of smaller bounding boxes is observed, suggesting that the model performs with higher accuracy on small to medium-sized objects in the vehicle's field of view, which is consistent with the typical sizes of objects encountered in traffic.

After training, the YOLOv8s model was evaluated using the test dataset, and the resulting confusion matrix is shown in Figure 4.3. This matrix illustrates the model's performance across different classes by displaying the rates of correct and incorrect classifications, thus highlighting its strengths and weaknesses. The confusion matrix indicates high accuracy for frequently represented classes (e.g., "car"), but higher error rates for underrepresented classes. This suggests that improving class balance could further enhance overall accuracy across all classes.

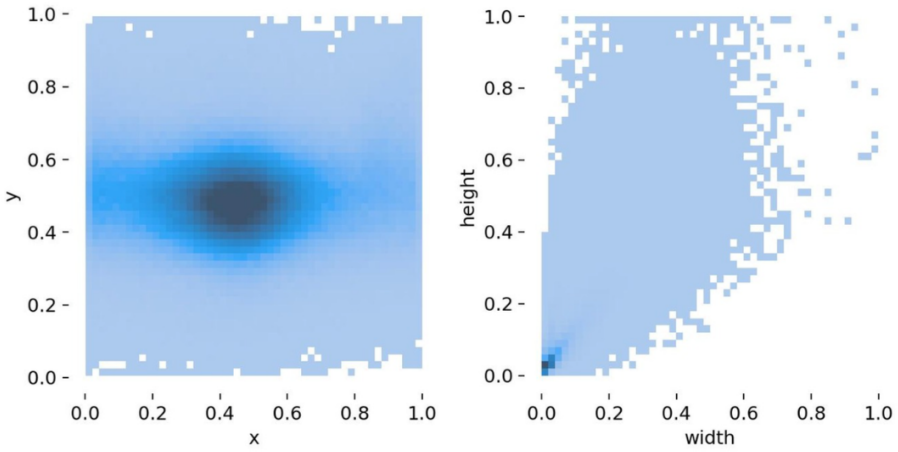

Figure 4.2 : Distribution of bounding box locations and sizes in the YOLOv8s model's dataset. The first plot displays the density of bounding box positions on the x-y coordinates, while the second plot illustrates the width and height distribution.

Overall, the performance metrics, dataset analyses, and evaluation results demonstrate the accuracy of both the object detection and depth estimation models, underscoring their potential contributions to real-time autonomous systems.

Figure 4.3 summarizes the model's performance across classes, emphasizing areas of high accuracy as well as those with misclassification. This analysis offers insights into areas where model improvement may be possible, guiding further enhancement of object detection accuracy for autonomous vehicle applications.

Figure 4.4 presents experimental results for both depth estimation and object detection, emphasizing the effectiveness of these models in object identification and distance estimation. Each image set includes a visualization of depth estimation alongside object detection outcomes, where depth is represented by a color gradient—lighter shades indicating closer objects and darker shades representing greater distances. This visualization provides a clear understanding of the spatial relationships between detected objects, offering valuable insights for autonomous navigation.

Figure 4.5 presents four example results for object detection and depth estimation. Each result includes visualizations of both object detection and depth estimation, with depth represented by a color gradient—lighter colors indicating closer objects and darker colors representing objects at greater distances.

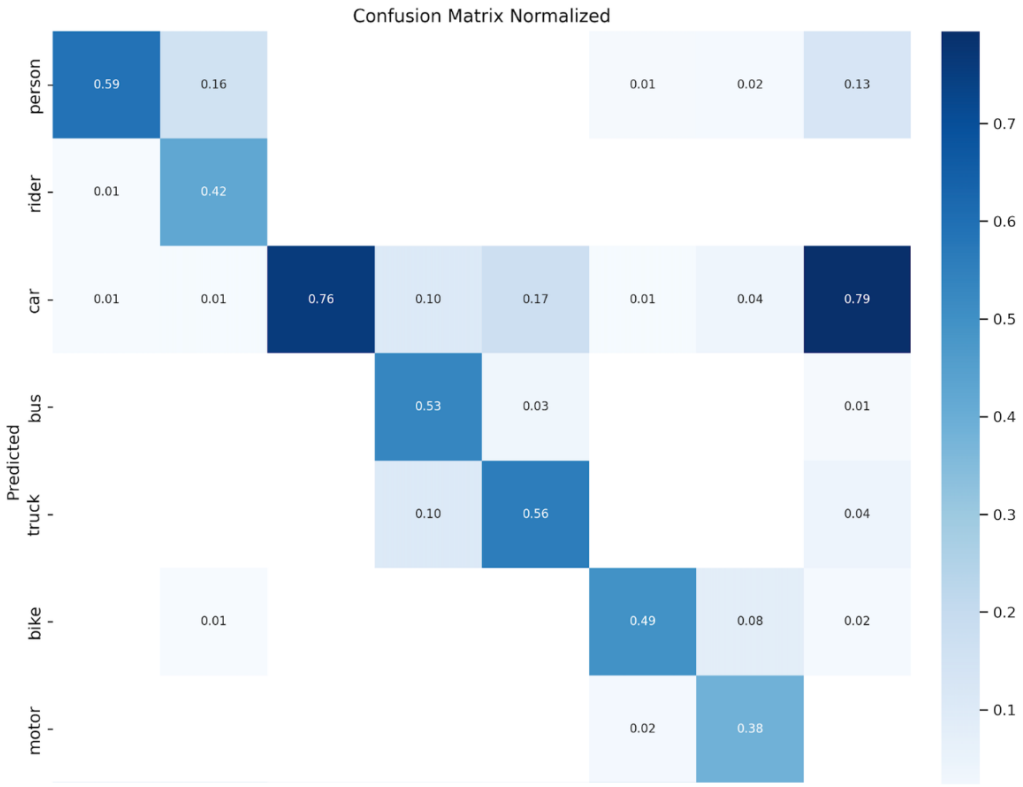

Figure 4.3 : Confusion matrix showing YOLOv8s model's classification performance on the test dataset, with correct classifications along the diagonal and misclassifications off-diagonal.

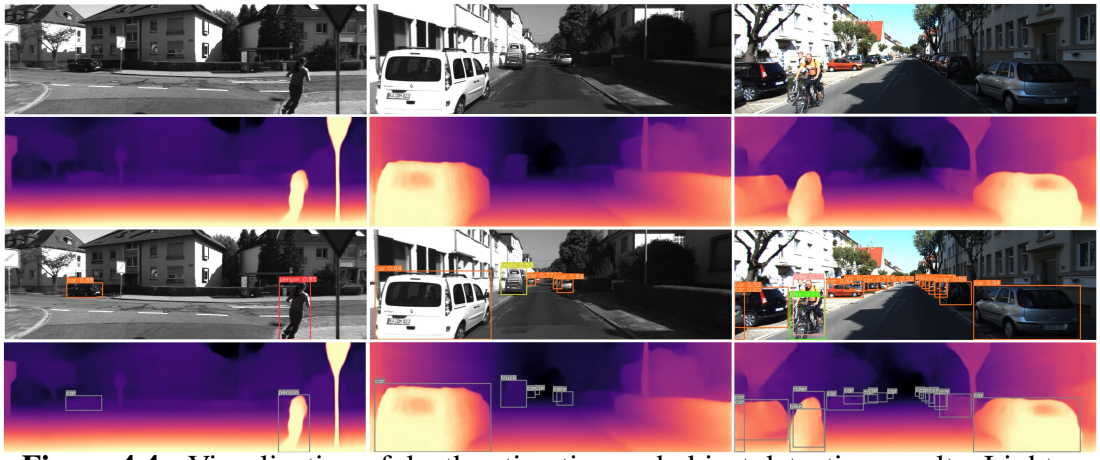

Figure 4.4 : Visualization of depth estimation and object detection results. Lighter colors denote closer objects, while darker colors represent more distant objects, aiding in the assessment of spatial positioning and object distance.

In the figure 4.4, the first row shows the original camera view, followed by the second row with overlays for depth estimation, and the third row with overlays for object detection.

Object detection is enhanced with bounding boxes and class labels, accurately detecting objects such as pedestrians, cars and cyclists. The integration of object detection with depth estimation significantly contributes to situational awareness in autonomous systems, as it enables both the recognition of object class and the estimation of object distance—key elements for real-time decision-making and path planning.

The depth map of the detected objects is extracted, and the relative distance estimation is calculated using this map. To evaluate the performance of the proposed system, a subset of 1,000 images was randomly selected from the KITTI Object Detection dataset. This dataset includes a variety of objects, such as cars, pedestrians, and motorcycles, and spans a range of distances. Importantly, these 1,000 images were excluded from the training process, ensuring an unbiased evaluation of the model's performance.

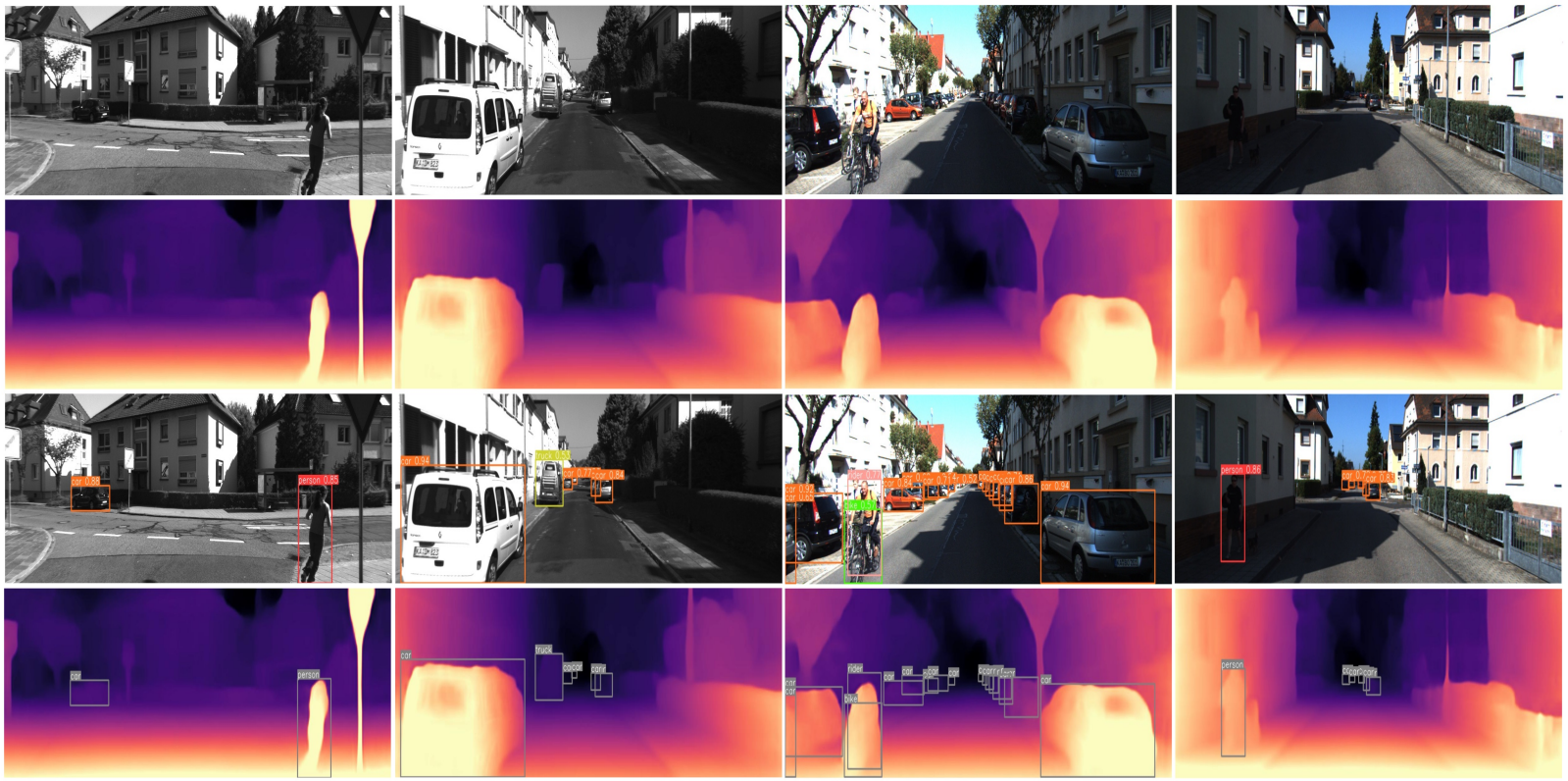

Figure 4.5 : Representative examples of object detection and depth estimation results.

The evaluation of the detection framework involved the use of accuracy and Root Mean Square Error (RMSE) as metrics to measure distance errors. Objects with accuracy values falling below the predefined threshold were excluded from consideration. RMSE served as a measure to quantify the error ratio between the predicted distance values and the corresponding actual distance values.

The depth maps of objects detected by YOLOv8 were extracted using Monodepth2. The combination of these two models was named "Model Combine (MC)" and subjected to a weighted average process with different ratios (R), denoted as Weighted Average (WAveg). Subsequently, Distance Estimation (DisEst) was performed, and the resulting MAE for distance is presented in the table. As shown in Table 4.1, it is evident that distance estimation obtained without fine-tuning on the KITTI object dataset is particularly satisfactory, especially for short distances. The table presents the performance of the model at close, medium, and far distances by providing distance values in three different measures. The weighted average algorithm, with a ratio of 7 and specifically for objects at distances of 0-10 meters, has demonstrated the best results.

Models	**R**	**0-10 m**	**10-25m**	**25+ m**
MC + WAveg	0	2.09	3.14	8.42
MC + WAveg + DisEst	5	1.76	3.01	8.07
MC + WAveg + DisEst	7	**1.21**	**2.43**	**5.42**
MC + WAveg + DisEst	15	4.45	3.58	8.72

Table 4.1 : Comparison of Distance Estimation Performance Across Different Model Combinations and Ratios (R). The table shows the Mean Absolute Error (MAE) in meters for objects at varying distances (0-10 m, 10-25 m, and 25+ m) using different model combinations and weight ratios (R).

Next, we evaluate our model performance with respect to FPS results in real-time, as shown in Figure 4.6. FPS values are measured on four different hardware configurations such as Server, Jetson-ONNX, Jetson-TensorRT-Float32 and Jetson-TensorRT-Integer8, respectively. Two key components (i) ONNX, an open format built to represent machine learning models, and (ii) TensorRT, an SDK specifically tailored for high-performance deep learning inference, are utilized during the evaluation.

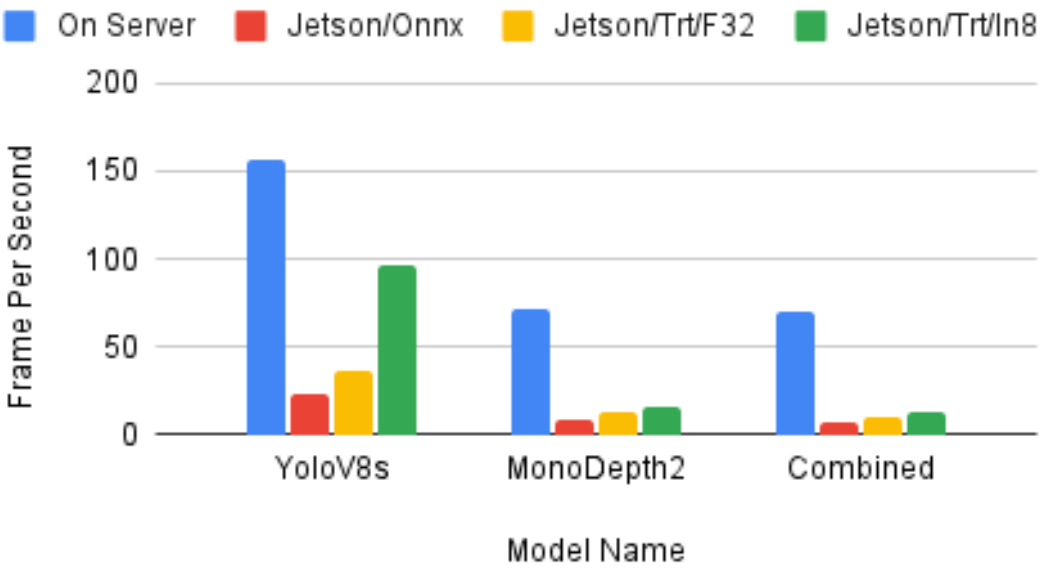

Figure 4.6 : Comparison of FPS performance for the YOLOv8s object detection model, Monodepth2 depth estimation model, and their combined usage across different hardware environments and optimization methods (Server with NVIDIA A100 GPU, Jetson device with ONNX, TensorRT float32, and TensorRT int8). The results illustrate the impact of hardware capabilities and optimization techniques on real-time applicability for autonomous vehicle applications.

The graph in Figure 4.6 illustrates the FPS performance of the developed object detection (YOLOv8s) and depth estimation (Monodepth2) models when deployed on different hardware environments and optimization methods. The primary aim of this study is to evaluate the performance of these models on the NVIDIA Jetson device to determine their suitability for real-time applications in autonomous vehicles.

The server environment, equipped with NVIDIA A100 PCIe GPUs and 30 GB of memory, provides high parallel processing power, making it suitable for computationally intensive deep learning models. The server achieves an FPS of approximately 150 for the YOLOv8s model, highlighting its capability to handle object detection at high speeds. The Monodepth2 model also performs well on the server, though it exhibits a lower FPS compared to YOLOv8s, likely due to the increased computational requirements associated with depth estimation.

On the Jetson device, performance evaluations were conducted using various optimization methods: ONNX format, TensorRT float32 (TrtF32), and TensorRT int8 (TrtIn8) conversions. The Figure shows that FPS values are relatively low in the

ONNX format on the Jetson, but significant improvements are observed with TensorRT optimizations. In particular, the int8 conversion notably enhances performance, reaching approximately 80 FPS for the YOLOv8s model. This finding suggests that int8 optimization is effective in improving efficiency on devices with limited computational resources. Similarly, the Monodepth2 model also benefits from the int8 conversion, achieving a higher FPS compared to the float32 conversion.

In the "Combined" scenario, where both models are run concurrently, a decrease in FPS is observed on both the server and Jetson device. This result, shown in Figure 4.6, indicates that running both models simultaneously imposes a considerable load on system resources, thereby reducing overall performance. Nevertheless, the int8 conversion continues to provide relatively better performance even in the combined mode on the Jetson.

These findings demonstrate that the optimized model achieves real-time operational feasibility on the Jetson device, particularly through int8 conversion. The observed performance enhancement with int8 conversion indicates that the model can meet the real-time processing and low-power consumption requirements of autonomous vehicle applications. In conclusion, this analysis supports the suitability of the developed model for practical deployment in autonomous driving systems, offering a viable level of performance for real-world use cases.

5. RUNNING ON IOT DEVICE

The real-time execution of tasks such as object detection and depth estimation in autonomous vehicles demands high computational power while simultaneously requiring reliable and cost-effective hardware solutions. In this context, deploying Internet of Things (IoT) devices with constrained processing power, memory capacity, and energy consumption for such applications underscores the importance of model optimization. This study aims to deploy the Monodepth2 depth estimation model and the YOLOv8 object detection model on IoT devices to enable surrounding object detection and depth estimation within autonomous vehicles.

To achieve this, model conversion processes were initially applied to adapt the models to the limited hardware capabilities of IoT devices. Subsequently, model quantization was conducted to minimize memory usage and reduce energy consumption. During the deployment and testing phases, the performance and accuracy of the models were then evaluated under real-world conditions. These processes constitute essential optimization steps to ensure that the Monodepth2 and YOLOv8 models operate efficiently and effectively within the constraints of autonomous vehicle applications.

5.1 Model Conversion and Quantization

In this study, conversion and optimization processes were applied to the YoloV8 and Monodepth2 models, aiming to reduce model size and improve inference speed for deployment on IoT devices. Initially, both models were converted from their native PyTorch framework to the ONNX format. ONNX provides platform independence, making it particularly well-suited for deployment on IoT and embedded systems. By converting to ONNX, the models shed complex data structures inherent in PyTorch, resulting in a more streamlined, computation-focused structure.

Following the ONNX conversion, further optimization was conducted using NVIDIA's TensorRT [38] library for both the YoloV8 and Monodepth2 models. The

TensorRT optimization process included reducing precision levels, such as converting from FP32 to FP16 or INT8, which effectively decreases memory usage while increasing inference speed. The INT8 quantization process, in particular, represents model weights and tensors in low precision, thereby enabling faster and more memory-efficient processing—a significant benefit for resource-constrained IoT devices. Additionally, TensorRT's dynamic tensor shape optimization was utilized to ensure the models operate within the memory constraints of the target devices. Optimization of CUDA cores by TensorRT further reduces computational load on IoT devices, accelerating the inference process.

To ensure successful inference on embedded systems, the hardware specifications and resource limitations of the target devices were carefully considered. For instance, INT8 and FP16 optimizations available in TensorRT were applied to models intended for deployment on NVIDIA Jetson platforms, which enabled enhanced performance and reduced memory consumption. This optimization process aligns the models' core operations with the capabilities of specific hardware, ensuring compatibility and stability across various embedded systems.

Furthermore, model quantization and pruning were explored to further reduce model size and complexity. Quantization enables low-precision calculations with minimal accuracy loss, a key advantage in IoT applications where both depth estimation and object detection demand reliable performance within limited computational resources. Pruning, on the other hand, retains only the essential network structure necessary for inference by removing redundant nodes, resulting in smaller models and faster inference times.

Converting from PyTorch to ONNX also improves inference time and memory usage. While PyTorch's dynamic computation graph offers advantages during model development, it can impose performance limitations during inference, especially in resource-constrained environments. By using ONNX's fixed computation graph, the models avoid unnecessary processing steps, thereby reducing computational load and memory consumption. This approach ensures faster inference, an essential enhancement for real-time applications in autonomous systems.

The process of converting the YoloV8 and Monodepth2 models from PyTorch to ONNX, followed by TensorRT optimization, results in models that are both compatible with embedded systems and highly efficient in terms of speed and memory usage. These optimizations make it feasible to deploy the models in critical real-time applications within autonomous vehicles, such as object detection and depth estimation.

5.2 Deployment and Testing on IoT Devices

The deployment of deep learning models on IoT devices entails several critical steps to ensure effective operation within the constraints of embedded systems. This study employed NVIDIA Docker for the deployment and testing processes of the YoloV8 and Monodepth2 models on NVIDIA Jetson Xavier NX platforms [39].

NVIDIA Docker is a robust tool that facilitates the deployment of containerized applications in environments utilizing NVIDIA GPUs [39]. It enables developers to create, manage, and run containers that leverage GPU acceleration. By encapsulating all dependencies, libraries, and runtime requirements within a Docker container, a consistent and reproducible environment is provided across various devices.

For the deployment of the models, Docker images were created that included the quantized versions of YoloV8 and Monodepth2, along with essential libraries such as TensorRT and OpenCV. These images were prepared using Dockerfiles that specify the base images and installation commands. Once the Docker images were prepared, they were uploaded to a container registry for efficient access.

On the target IoT device, the container image was pulled and executed using NVIDIA Docker. The lightweight nature of containers allows for rapid deployment of models, circumventing the typical overhead associated with traditional virtual machines. The hardware acceleration capabilities of the Jetson Xavier NX yield optimized inference speeds, which are critical for real-time applications such as object detection and depth estimation [40].

Following deployment, rigorous testing was conducted to validate the performance of the models on IoT devices. These tests involved evaluating the inference speed,

memory usage, and accuracy of the models under various scenarios. Sample datasets were utilized to simulate real-world conditions and assess the models' adaptability to hardware constraints.

By leveraging NVIDIA Docker for deployment, the process of executing deep learning models on IoT devices was streamlined. This approach not only simplifies the deployment workflow but also ensures that the models operate effectively within the limitations of embedded systems. The testing phase confirmed that the quantized models were capable of performing object detection and depth estimation tasks efficiently, thereby demonstrating their viability for use in autonomous systems and other applications requiring real-time processing on resource-constrained devices.

6. CONCLUSIONS

6.1 Summary and Contributions

In this thesis, a dual deep learning framework is proposed, combining a supervised model for object detection and a self-supervised model for depth estimation. For object detection, the YOLOv8 model is chosen due to its superior accuracy and FPS performance compared to earlier versions. The model is trained using selected object classes from the BDD100K dataset. For depth estimation, the Monodepth2 model is employed, trained on monocular video sequences from the KITTI dataset. The detected objects are projected onto the depth estimation map to calculate their distances. One of the key challenges addressed in this study is the discrepancy between the object and background depths within the bounding boxes during object detection. A novel algorithm is developed to mitigate this error, ensuring more accurate distance measurements.

In addition to the development of the deep learning models, this work also focuses on real-time application. The YOLOv8 and Monodepth2 models are integrated on the Nvidia Jetson Xavier NX platform, leveraging optimized GPU performance and the TensorRT library for efficient processing in real-time environments.

The main contributions of this thesis are as follows:

The YOLOv8 model is utilized for object detection, offering a high detection accuracy with low latency, making it suitable for real-time applications. Its performance is thoroughly validated through extensive evaluation on the BDD100K dataset.

A novel approach is proposed to address the error caused by background depth within the object bounding boxes during object detection. A weighted average of depth values is used to accurately reflect the depth of the detected objects, significantly improving distance estimation precision.

A Linear Regression model is trained using the median RGB channel values extracted from the weighted depth map. This approach enhances the robustness of depth estimation, resulting in more reliable outputs.

The integration of the YOLOv8 and Monodepth2 models is successfully implemented on the Nvidia Jetson Xavier NX platform. By utilizing GPU optimization and the TensorRT library, real-time performance is achieved, demonstrating the system's effectiveness in practical applications.

6.2 Future Work and Limitations

This study introduces a dual deep learning architecture designed for real-time object detection and depth estimation. Despite the promising results achieved, there are several limitations and potential areas for enhancement that should be considered in future research.

Firstly, the Monodepth2 model, employed for depth estimation, is exclusively trained on monocular images. This may restrict its performance, particularly in complex scene structures or scenarios where more advanced sensing technologies such as stereo cameras or LiDAR are necessary for accurate depth perception. Future research could investigate the integration of more diverse data sources, such as stereo images or LiDAR, to improve depth estimation accuracy. Moreover, the Monodepth2 model was trained using only the KITTI dataset, which may hinder its generalizability across different environmental conditions (e.g., nighttime, adverse weather such as rain). Expanding the training dataset to encompass a wider variety of environmental conditions would likely enhance the model's robustness and adaptability.

Regarding object detection, although the YOLOv8 model delivers fast and accurate results, it may face challenges in detecting very small or partially occluded objects. Future work could explore more sophisticated object detection models or consider retraining the YOLOv8 model to better handle such challenging scenarios. Furthermore, to enhance the precision of depth and distance measurements for objects of varying sizes, additional deep learning algorithms could be developed to incorporate object size more effectively in the depth estimation process.

A further limitation pertains to the system's performance on the Nvidia Jetson Xavier NX platform. Although optimized for this hardware, the system's performance may degrade in more complex or large-scale scenes. To address this, future research could focus on utilizing more powerful hardware platforms or applying model compression techniques to maintain efficiency while ensuring real-time performance on existing hardware.

Lastly, the models and algorithms developed in this study have been evaluated on specific datasets. To ensure broader applicability and improve generalization, future research should involve testing across a more diverse set of scenarios and datasets. This would provide stronger validation and facilitate the system's deployment in real-world applications.

6.3 Conclusion

In this work, we presented a new approach addressing the problem of depth estimation from a single monocular camera image in real-time. Unlike the use of stereo videos or expensive sensors, our method only necessitates a single monocular camera image and delivers real-time responses, making it suitable for computation-limited embedded devices. Moreover, we illustrate a more accurate approach to depth estimation calculation by reducing background image error. As the future work, we plan to incorporate segmentation on bounding boxes for robust depth estimation. This approach can help prevent occlusion issues. We use Nvidia Jetson Xavier NX in this study and we plan to upgrade the embedded device to achieve better FPS results. In addition, Monodepth2 layers are not fully compatible with TensorRT SDK. If TensorRT becomes fully compatible with new versions, we would like to update the layers accordingly.

REFERENCES

[1] **Lu, Y. and Lu, G.** (2021). An alternative of lidar in nighttime: Unsupervised depth estimation based on single thermal image, *Proc. of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp.3833–3843.

[2] **Shi, P.**, **Peng, J.**, **Qiu, J.**, **Ju, X.**, **Lo, F.P.W. and Lo, B.** (2023). EVEN: An Event-Based Framework for Monocular Depth Estimation at Adverse Night Conditions, *arXiv preprint arXiv:2302.03860*.

[3] **Casser, V.**, **Pirk, S.**, **Mahjourian, R. and Angelova, A.** (2019). Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos, *Proc. of the AAAI conference on artificial intelligence*, pp.8001–8008.

[4] **Masoumian, A.**, **Marei, D.G.**, **Abdulwahab, S.**, **Cristiano, J.**, **Puig, D. and Rashwan, H.A.** (2021). Absolute Distance Prediction Based on Deep Learning Object Detection and Monocular Depth Estimation Models., *CCIA*, pp.325–334.

[5] **Tosi, F.**, **Aleotti, F.**, **Poggi, M. and Mattoccia, S.** (2019). Learning monocular depth estimation infusing traditional stereo knowledge, *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.9799–9809.

[6] **Li, Z.**, **Liu, X.**, **Drenkow, N.**, **Ding, A.**, **Creighton, F.X.**, **Taylor, R.H. and Unberath, M.** (2021). Revisiting stereo depth estimation from a sequence-to-sequence perspective with transformers, *Proceedings of the IEEE/CVF international conference on computer vision*, pp.6197–6206.

[7] **Godard, C.**, **Mac Aodha, O. and Brostow, G.J.** (2017). Unsupervised monocular depth estimation with left-right consistency, *Proc. of the IEEE conference on computer vision and pattern recognition*, pp.270–279.

[8] **Shamsafar, F.**, **Woerz, S.**, **Rahim, R. and Zell, A.** (2022). Mobilestereonet: Towards lightweight deep networks for stereo matching, *Proceedings of the ieee/cvf winter conference on applications of computer vision*, pp.2417–2426.

[9] **Howard, A.G.** (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications, *arXiv preprint arXiv:1704.04861*.

[10] **Sandler, M.**, **Howard, A.**, **Zhu, M.**, **Zhmoginov, A. and Chen, L.C.** (2018). Mobilenetv2: Inverted residuals and linear bottlenecks, *Proceedings*

of the IEEE conference on computer vision and pattern recognition, pp.4510–4520.

[11] **Author, A. and Author, B.** (2022). A Comprehensive Review of Object Detection Techniques, *Journal of Computer Vision*, *123*(4), 1–25.

[12] **Ren, S.**, **He, K.**, **Girshick, R. and Sun, J.** (2016). Faster R-CNN: Towards real-time object detection with region proposal networks, *IEEE transactions on pattern analysis and machine intelligence*, *39*(6), 1137–1149.

[13] **Liu, W.**, **Anguelov, D.**, **Erhan, D.**, **Szegedy, C.**, **Reed, S.**, **Fu, C.Y. and Berg, A.C.** (2016). Ssd: Single shot multibox detector, *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, Springer, pp.21–37.

[14] **Jocher, G.**, **Qiu, J. and Chaurasia, A.** (2023). *Ultralytics YOLO*, `https://github.com/ultralytics/ultralytics`.

[15] **Lin, T.**, **Maire, M.**, **Belongie, S.J.**, **Bourdev, L.D.**, **Girshick, R.B.**, **Hays, J.**, **Perona, P.**, **Ramanan, D.**, **Doll'a r, P. and Zitnick, C.L.** (2014). Microsoft COCO: Common Objects in Context, *CoRR*, *abs/1405.0312*, `http://arxiv.org/abs/1405.0312`, `1405.0312`.

[16] **Ronneberger, O.**, **Fischer, P. and Brox, T.** (2015). U-net: Convolutional networks for biomedical image segmentation, *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, Springer, pp.234–241.

[17] **Godard, C.**, **Mac Aodha, O.**, **Firman, M. and Brostow, G.J.** (2019). Digging into self-supervised monocular depth estimation, *Proceedings of the IEEE/CVF international conference on computer vision*, pp.3828–3838.

[18] **He, K.**, **Zhang, X.**, **Ren, S. and Sun, J.** (2016). Deep residual learning for image recognition, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp.770–778.

[19] **Godard, C.**, **Mac Aodha, O. and Brostow, G.J.** (2017). Unsupervised monocular depth estimation with left-right consistency, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp.270–279.

[20] **Kendall, A.**, **Grimes, M. and Cipolla, R.** (2016). *PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization*, `https://arxiv.org/abs/1505.07427`, `1505.07427`.

[21] **He, K.**, **Zhang, X.**, **Ren, S. and Sun, J.** (2015). *Deep Residual Learning for Image Recognition*, `https://arxiv.org/abs/1512.03385`, `1512.03385`.

[22] **Wang, T.**, **Pang, J. and Lin, D.** (2022). Monocular 3d object detection with depth from motion, *European Conference on Computer Vision*, Springer, pp.386–403.

[23] **developers, O.R.** (2021). *ONNX Runtime*, `https://onnxruntime.ai/`, version: 18.1.0.

[24] **Yu, F.**, **Xian, W.**, **Chen, Y.**, **Liu, F.**, **Liao, M.**, **Madhavan, V.**, **Darrell, T.** *et al.* (2018). Bdd100k: A diverse driving video database with scalable annotation tooling, *arXiv preprint arXiv:1805.04687*, *2*(5), 6.

[25] **Geiger, A.**, **Lenz, P.**, **Stiller, C. and Urtasun, R.** (2013). Vision meets Robotics: The KITTI Dataset, *International Journal of Robotics Research (IJRR)*.

[26] **Bray, T.** (2014). *JavaScript Object Notation (JSON)*, RFC 7159, `https://www.ietf.org/rfc/rfc7159.txt`.

[27] **Eigen, D.**, **Puhrsch, C. and Fergus, R.** (2014). Depth map prediction from a single image using a multi-scale deep network, *Advances in neural information processing systems*, *27*.

[28] **Wang, C.Y.**, **Liao, H.Y.M.**, **Wu, Y.H.**, **Chen, P.Y.**, **Hsieh, J.W. and Yeh, I.H.** (2020). CSPNet: A new backbone that can enhance learning capability of CNN, *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp.390–391.

[29] **Tan, M.**, **Pang, R. and Le, Q.V.** (2020). Efficientdet: Scalable and efficient object detection, *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp.10781–10790.

[30] **Shi, Q.**, **Wang, D.**, **Chen, W.**, **Yu, J.**, **Zhou, W.**, **Zou, J. and Liu, G.** (2023). Research on Spaceborne Target Detection Based on Yolov5 and Image Compression, *Future Internet*, *15*(3), 114.

[31] **Goodfellow, I.** (2016). *Deep learning*.

[32] **Zheng, Z.**, **Wang, P.**, **Liu, W.**, **Li, J.**, **Ye, R. and Ren, D.** (2020). Distance-IoU loss: Faster and better learning for bounding box regression, *Proceedings of the AAAI conference on artificial intelligence*, pp.12993–13000.

[33] **Li, X.**, **Wang, W.**, **Wu, L.**, **Chen, S.**, **Hu, X.**, **Li, J.**, **Tang, J. and Yang, J.** (2020). Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection, *Advances in Neural Information Processing Systems*, *33*, 21002–21012.

[34] **Garg, R.**, **Bg, V.K.**, **Carneiro, G. and Reid, I.** (2016). Unsupervised cnn for single view depth estimation: Geometry to the rescue, *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, Springer, pp.740–756.

[35] **Zhou, T.**, **Brown, M.**, **Snavely, N. and Lowe, D.G.** (2017). Unsupervised learning of depth and ego-motion from video, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp.1851–1858.

[36] **Ronneberger, O.**, **Fischer, P. and Brox, T.** (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*, `https://arxiv.org/abs/1505.04597`, `1505.04597`.

[37] **Deng, J.**, **Dong, W.**, **Socher, R.**, **Li, L.J.**, **Li, K. and Fei-Fei, L.** (2009). Imagenet: A large-scale hierarchical image database, *2009 IEEE conference on computer vision and pattern recognition*, Ieee, pp.248–255.

[38] **NVIDIA** (2023). *TensorRT: High Performance Deep Learning Inference Library*, `https://developer.nvidia.com/tensorrt`.

[39] **Corporation, N.** (2019). *NVIDIA Docker: A Tool for Deployment of GPU-Accelerated Applications*, `https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html`.

[40] **Corporation, N.** (2021). *NVIDIA Jetson Xavier NX: The Platform for AI and Deep Learning*, `https://developer.nvidia.com/embedded/jetson-xavier-nx`.

CURRICULUM VITAE

Name SURNAME: Emre ÇETİN

EDUCATION:

- **B.Sc.:** 2019, Yildiz Technical University, Faculty of Computer and Informatics Engineering, Computer Engineering Department
- **M.Sc.:** 2024, Istanbul Technical University, Department of Computer Engineering, Computer Engineering Programme

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2021-Current, Artificial Intelligence Software Engineer at Turkcell Technology
- 2020-2021, Artificial Intelligence Software Engineer at Baykar
- 2019-2020, Software Engineer at Softtech

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Çetin, E.**, Sarl, T. T., Assoy, M., Seçinti, G. (2024, November). Monodepth-Based Object Detection and Depth Sensing for Vehicle Vision Systems. *In 2024 IEEE 10th World Forum on Internet of Things (WF-IoT)* (pp. 696-701). IEEE.
- E. E. Aydın, **E. Çetin**, T. T. Sarı, G. Seçinti, From Theory to Practice: A Testbed for DL-Based Semantic Communication Architecture. *2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC), Las Vegas, USA, January 2025.*