Depth-Aware Object Detection and Region Filtering for Autonomous Vehicles: A Monocular Camera-Based Novel Integration of MiDaS and YOLO for Complex Road Scenarios with Irregular Traffic

by

Md. Imamul Mursalin Sujoy
20301120
Md. Asif Rahman
20301122
Utsha Sen Dhruba
20301338
Sartaj Emon Prattoy
20301326

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
October 2024

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Imamul M. Sujoy	Asif Rahman
Md. Imamul Mursalin Sujoy 20301120	Md. Asif Rahman 20301122
Utsha Sen	[illegible]
Utsha Sen Dhruba 20301338	Sartaj Emon Prattoy 20301326

Approval

The thesis/project titled "Depth-Aware Object Detection and Region Filtering for Autonomous Vehicles: A Monocular Camera-Based Novel Integration of MiDaS and YOLO for Complex Road Scenarios with Irregular Traffic" submitted by

1. Md. Imamul Mursalin Sujoy(20301120)
2. Md. Asif Rahman(20301122)
3. Utsha Sen Dhruba(20301338)
4. Sartaj Emon Prattoy(20301326)

Of Summer, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on October, 2024.

Examining Committee:

Supervisor:
(Member)

Md. Khalilur Rhaman

Dr. Md Khalilur Rhaman
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

In this era of technological advancement, autonomous vehicles strive towards revolutionizing the transportation system. Object detection and distance measurement are critical for autonomous driving, particularly in dynamic and unpredictable environments. This paper presents a new approach for improving object detection and depth estimation in autonomous vehicles, with a focus on the complex road conditions of Bangladesh. Using MiDaS for depth estimation and YOLOv8 for object detection, we introduced the Depth Aware ROI Filter (DARF) to refine the region of interest, enhancing detection accuracy while reducing processing time by more than 85%. By focusing solely on depth-relevant areas, our method optimizes object detection, combining depth information and object classification essential for autonomous navigation. Tested on a diverse dataset of Bangladeshi road scenarios, including various vehicles and weather conditions, our approach ensures real-world applicability. Additionally, we explored several regression models to calculate the relative depth-to-real distance scale factor, further improving depth accuracy. This work demonstrates the potential of our system to enhance real-time decision-making in autonomous vehicles, paving the way for advancements in object tracking and system optimization in complex road environments.

Keywords: MiDaS; ROI; Relative; Detection; Integration; YOLO; Regression Analysis

Dedication

We would like to dedicate this thesis to our beloved parents, siblings and friends for continuous support, love, and motivation throughout our undergraduate study. Their belief, sacrifice, and motivation toward this milestone proved invaluable. Without their consistent level of presence and inspiration, this work would not have been possible. Indeed, this is the reflection of their timeless support and strength that they have instilled in us.

Acknowledgement

Firstly, all praise to the Great Almighty for whom our thesis have been completed without any major interruption.

Secondly, to our supervisor Dr. Md. Khalilur Rhaman sir for his kind support and advice in our work.

And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

DARF	Depth-Aware Region Filter
MiDaS	Multiple Depth Estimation Accuracy with Single Network
PR	Precision-Recall
ROI	Region of Interest
YOLO	You Only Look Once

Chapter 1

Introduction

In recent years, autonomous vehicles have received huge public attention for their potential to change transportation and alter the way people move within urban environments. This can offer safer, more efficient travel by reducing human error, smoothing traffic flow, and mobilizing all. But one of the critical success factors is their capability for correct real-time obstacle detection and motion in complex environments. Object detection and depth sensing are some of the most salient aspects of real-world driving in dynamic and unpredictable conditions, especially in a densely populated area. An autonomous vehicle has to detect not only static and moving objects but also understand the relative distances from the vehicle for informed decisions on speed, braking, and steering. These challenges are accentuated in the context of countries like Bangladesh, where the highly invariable road conditions coupled with constant monitoring for everything from pedestrians and bicycles to cars and rickshaws will further exacerbate these challenges.

Our research fuses MiDaS-based depth estimation with object detection and introduces a novelty of Depth Aware ROI Filter (DARF) that enhances the process of detection. This approach focuses on enhancing the vehicle's decision-making ability by prioritizing the regions of interest by ranking them according to their proximity and applying dynamic filtering to remove those parts of the scene that are irrelevant. In this regard, we are not only aiming at improving the accuracy and speed of detection but also trying to provide important depth information that can be used effectively for path planning, navigation, and tracking the movement of moving objects. This will open the path to even smarter autonomous systems that can manage complex urban scenarios much more efficiently, thus bringing us closer to getting the full benefits of autonomous vehicles.

1.1 Research Problem

Our purpose of the paper is to prepare a method for autonomous vehicle vision according to Bangladesh's condition and geographical location. Several considerations make it difficult to introduce autonomous cars in Bangladesh. The unpredictable and difficult driving environment is a result of the country's crowded and disorganized road infrastructure and the frequent lack of respect to traffic laws. In order to function properly, autonomous cars depend on organized and rule-based systems, which goes against the present norms of driving in Bangladesh. The absence of standardized signs, road markings, and road capacity in the country only adds to the challenges of creating and implementing autonomous vehicle technology. There's no denying that Bangladesh needs to work on autonomous cars. Possible benefits include making roads safer, decreasing traffic, and increasing productivity in the transportation sector. Moreover, Bangladesh's harsh monsoon rains, blistering summer temperatures, and occasional fog all pose significant challenges to the widespread use of autonomous vehicles. The effectiveness of sensors, driver sight, and the general safety of autonomous systems may all be affected by such weather conditions. For that reason, we want to build an autonomous vehicle object depth sensing system using cameras that would work in the aforementioned conditions.

For the depth sensing of the surrounding environment in autonomous vehicles, multiple methodologies are followed; all these bear certain limitations. Among such is LiDAR, certainly one of the most acknowledged technologies today that estimates depth by sending pulses of laser light onto objects. As people often say, LiDAR has extreme accuracy when it comes to depth estimation, but also has a number of huge disastrous disadvantages. First and foremost, the costliness of LiDAR contributes a lot to making it impracticable for applications in consumer-grade vehicles[14]. Also, LiDAR systems are large and have to be placed with much care, hampering vehicle design and aerodynamics. Another limitation with LiDAR is that it is prone to meteorological interference whereby rain, fog, and snow could impede its functionalities and, thus, the accuracy of depth readings. Besides, LiDAR has very excessive power consumption, a fact that has negative effects directly on vehicle efficiency, especially for electric vehicles. The LiDAR technique also faces difficulties when working with reflective and transparent surfaces, as the latter may distort the data about depth.

Other methods involved tend to have more weaknesses, such as stereo vision. In stereo vision, two cameras mounted side by side would use image comparison to create depth perception. While it can produce highly accurate 3D data, this method is bound to have errors due to either the movement of the vehicle or that of the object captured. Stereo vision is also sensitive to reflective surfaces, and it has a range of effectiveness, making it less usable for real-time decision-making in dynamic environments[1]. In occlusions, the performance is bad, and mostly, it fails to represent smooth surfaces with equally detailed information. The stereo cameras are complex to calibrate, and the real processing computationally demanding makes them even less effective. In contrast, our proposed approach fuses the capabilities of MiDaS for depth sensing with YOLO for object detection and enhances the process even further by incorporating a Depth Aware ROI Filter. By implementing depth filtering, our system skips the processing of unnecessary parts and targets the relevant regions of interest, enhancing the efficiency of detection by more than 85% and making our method nearly seven times faster when compared to traditional object detection over the full image. Unlike LiDAR and stereo vision, ours does not require any special hardware but can work with ordinary RGB cameras, which is cost-effective and easy to deploy in a wide variety of vehicle platforms. Besides, compared to LiDAR and stereo vision, our method is far more robust in normal scenarios where reflective or smooth surface conditions exist.

1.2 Research Objective

It is an innovative and demanding ambition to create an autonomous car that can adapt to the distinct road and weather conditions of Bangladesh. Our respected supervisor's idea inspired this research paper, which inspired us towards solving regional transportation problems. We find it fascinating that the vision system of autonomous vehicles where it can detect objects precisely and efficiently is in Bangladesh's perspective. The research paper's emphasis on using monocular cameras alone for depth sensing and object recognition is an interesting central goal. This comprehensive approach to developing autonomous vehicles is shown by the use of monocular camera setup to handle the wide variety of conditions that might arise in the dense traffic of Bangladesh. This strategy prioritizes efficiency, economy, and longevity, all of which may be seen in its efforts to use pre existing infrastructure. Some of the notable objectives of this research are:

1. To provide an efficient and cost friendly object detection and depth sensing system for autonomous vehicles using only monocular camera setup.

2. To provide a method that works best for Bangladesh's specific requirements, demonstrating the viability of long-term, cost-effective solutions and expanding the frontiers of computer vision and image processing.

3. To reduce the complexity of object depth sensing of autonomous vehicles where no modal setup would be used.

4. To evaluate the monocular camera depth estimation and object detection integration method by using it in different scenarios and weather conditions of Bangladesh which enables for future development.

5. To offer recommendations on improving the integration method which would be left for future work in this field.

Chapter 2

Literature Review and Related Work

Demonstrated in the last few years, deep learning performed superior in a wide variety of computer vision applications compared to traditional computer vision algorithms. And it improved the speed and precision of processing for deep learning models. Chang et al. in their paper[13] propose a new approach of depth estimation with a single camera by incorporating deep optics. The authors introduce the end design of optics and image processing for monocular depth estimation. Their approach leverages depth-coding through coded defocus blur and chromatic aberrations in varying wavelengths, which are then decoded using a CNN. This is actually a good approach to the problem since most of the monocular depth estimation methods rely on recovering 3D information that was lost during capture. The authors employ deep optics to perform optimization methods such as freeform lens design and show how optical aberrations can actually help improve depth estimation. Specifically, it is found that the combination of optimized lens design and CNNs improves depth estimation and 3D object detection results even more than traditional methodologies of all-in-focus image-based approaches.

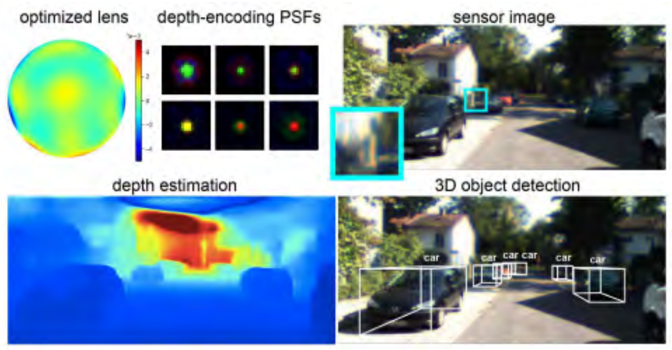


Figure 2.1: End-to-end design of optics and image processing, to build an optical-encoder[13]

Extensive experiments have been carried out by the authors on datasets such as NYU Depth v2 and KITTI to validate their approach by showing that their system is performing well in various environments. They further create a physical prototype to test their method on real-world data and show that chromatic aberrations are helping significantly with depth estimation. They also demonstrate that improved depth estimation indeed benefits 3D object detection, especially when the optimized lens setup is used. Their findings indicated that jointly optimizing camera optics along with neural networks could be valuable not only for improving depth perception but also for higher-level tasks such as object detection.

In another paper approach toward fast and robust monocular depth estimation for obstacle detection in high-speed autonomous systems. Authors Mancini et al., in the paper[4] introduce another approach for fast and efficient monocular depth estimation in obstacle detection in high-speed autonomous systems. The main novelty in this work is the use of a fully convolutional DNN, whereby the network takes as input monocular images and optical flow to estimate depth.

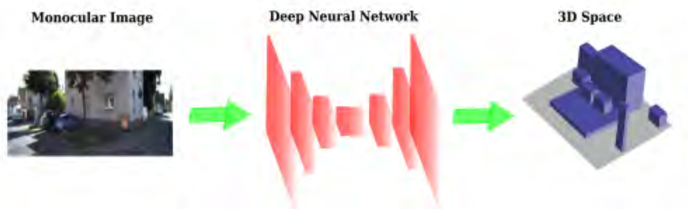


Figure 2.2: Fully convolutional network fed with both images and optical flows to obtain fast and robust depth estimation[4]

Their approach was trained with both real and synthetic data, considering photorealistic outdoor environments generated by the use of a state-of-the-art graphics engine. Merging real-world data with synthetic sequences from the dataset overcomes the limitations imposed by traditional training datasets and helps generalize the model better across diverse real-world conditions. The results show that their approach can estimate depth at high speed, about 300Hz, hence it is suitable for applications with fast motion, like autonomous driving. The authors further test the robustness of their model by introducing artificial blur and lighting variations and show that the network is able to make good performance under noisy conditions. Their experiments, which they ran on the KITTI dataset, further validate the effectiveness of their approach, where their system achieves competitive results comparable to the state-of-the-art methods without even fine-tuning, using synthetic data for training.

Furthermore, the authors introduce in this work a new, efficient method for real-time monocular depth estimation on embedded systems in the paper[16]. The work focuses on the solving of one of the major challenges with robotics and autonomous systems, where lightweight and efficient depth estimation models are at premium due to their limited computational resources. They introduce a fully convolutional encoder-decoder network architecture that utilizes MobileNet as an encoder; this would effectively extract depth information from a single RGB image. They made use of depth wise separable convolutions and nearest neighbor interpolation in the decoder to achieve significant reductions in complexity and latency.

Moreover, they further apply network pruning by the NetAdapt algorithm, iteratively removing the redundant layers and significantly reducing the total number of

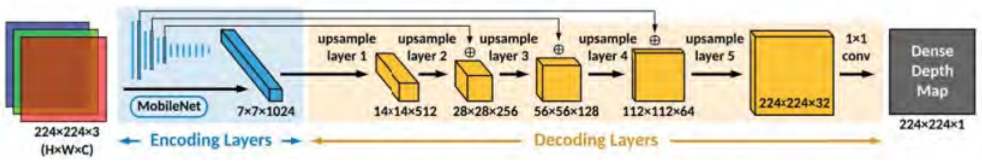
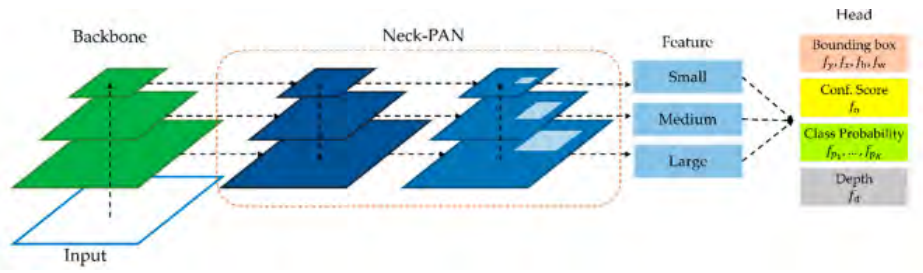


Figure 2.3: FastDepth architecture[16]

parameters with a slight degradation cost. The proposed model, FastDepth, runs on an NVIDIA Jetson TX2 GPU at 178 fps, sacrificing only a marginal amount of accuracy compared to state-of-the-art depth estimation models. Via careful design optimizations, the paper shows that FastDepth achieves an order of magnitude higher speeds compared to existing models while yielding state-of-the-art accuracies on the NYU Depth v2 dataset.

Following this, the paper[22] by Jongsub Yu and Hyukdoo Choi proposes a method of object detection integrated with depth estimation using monocular camera images. It investigates the simplification of depth prediction for every detected object to support better risk assessments for the autonomous driving system. Unlike those 3D object detection models that require either complex outputs or are based on resource-intensive LiDAR point clouds, the described approach simplifies the network by embedding depth estimation directly into a 2D object detector (YOLOv4).



Backbone
Neck-PAN
Feature
Head
Small
Medium
Large
Bounding box f_y, f_x, f_h, f_w
Conf. Score f_o
Class Probability $f_{p_1}, \dots, f_{p_K}$
Depth f_d
Input

Figure 2.4: YOLO MDE architecture[22]

The idea includes adding a channel to the fully-featured YOLOv4 output layer for detecting the depth of objects detected in 2D. Depth information is extracted from the 3D bounding box coordinates provided from the ground truth labels in the dataset. By using the closest depth from the 3D bounding box, the network will be trained to predict depths effectively. The authors compare their model with state-of-the-art 3D detection models using the benchmark dataset KITTI. They claim to have improved performance in detection and speed of the approach to fit real-time applications in autonomous driving systems. By doing so, this work proves that per-object single-depth estimation is sufficient, rather than the more complicated 3D bounding box estimation necessary for depth information to assess the risk in autonomous driving and further promote computational efficiency.

In another study[20], Masoumian et al. propose a new framework that predictively estimates the absolute distance of objects in a scene using 2D images by combining monocular depth estimation and object detection knowledge. The system uses two parallel deep learning networks: one for object detection based on YOLOv5, and another for depth estimation based on a self-supervised deep autoencoder network.

While the YOLOv5 network executes object detection and localization with bounding boxes, the depth estimation network calculates the relative distances of objects concerning the position in the depth map. Later, these relative distances will help to compute the absolute distance of the objects through a quadratic equation that was calibrated with real-world data.

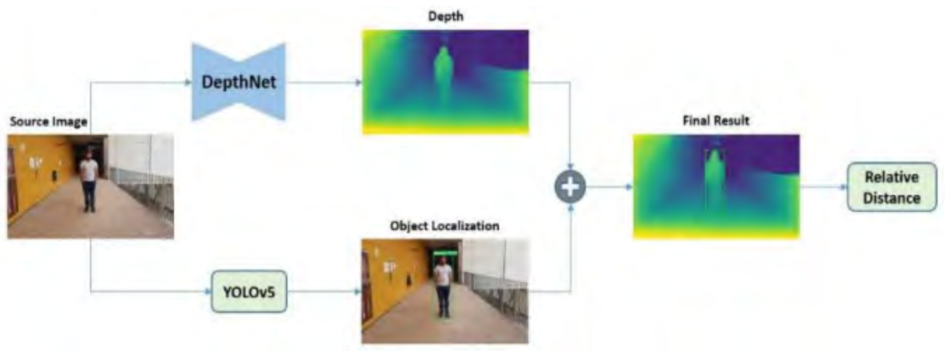


Figure 2.5: Fusing DepthNet and YOLOv5 for absolute distance[20]

This is an important approach for autonomous systems, as for example, distance prediction from a monocular camera is not as costly and is much more productive compared to LiDAR or stereo vision electronic setups. The model performs highly accurate performances with 96 percent accuracy with a root mean square error of 0.203 meters for absolute distance prediction. It also has a great possibility of finding its application in extended applications such as autonomous driving, for which real-time, low-cost distance estimation is locally critical. Future work will involve the model in both the detection of smaller objects that are further away and the further refinement of the distance prediction algorithm.

In the paper[17], the authors have provided a thorough examination of depth sensing techniques, classifying them into active and passive approaches. Passive techniques include multi-view object identification and monocular vision approaches, while active methods include interaction with objects using ultrasonic and Time-of-Flight (ToF) sensors. This research analyzes the efficacy of several supervised models for accurate depth estimation on the KITTI dataset. These models include the Attention-Based Context Aggregation Network, CNN, Spacing-Increasing Discretization (SID), 3D geometric constraints using virtual norms (VNs), Local Planar Guidance Layers (LPGLs), and DeepV2D. The paper also delves into semi-supervised methods like GASDA and investigates self-supervised models such Siamese DispNet, ResNet, and VGG, providing invaluable insights into the landscape of depth sensing techniques and associated performance measures. Finally it has shown they have reviewed different types of models. In the KITTI dataset DeepV2D gives more accurate results.

The work in the paper[8] by Tateno et al. introduces a novel approach that combines predicted depth maps from a CNN with depth measurements from traditional monocular SLAM. This yields an accurate, real-time scene reconstruction. The methodology utilizes CNN-predicted depth maps to remedy deficiencies in the performance of monocular SLAM in areas where SLAM tends to fail, such as low-textured regions. The CNN-based depth prediction gives an initial estimation of the depth map, then it gets refined by small-baseline stereo matching to enhance local details.

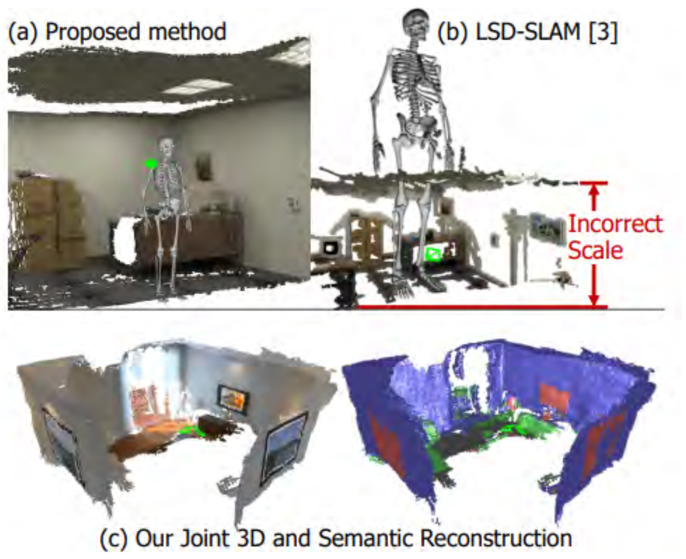


Figure 2.6: 3D and semantic reconstruction from a single view[8]

The key contribution of this paper is the estimation of the absolute scale of the scene, thereby overcoming one of the major drawbacks of SLAM. The use of deep learning for the prediction of depth in this approach makes it certain that even in places where the tracking of SLAM may be poor, dependable depth information will be observed. Also, integration of semantic segmentation into the SLAM framework provides an avenue for the authors to create semantically meaningful scene reconstructions from monocular input. A combination of SLAM and learned depth maps allows the system to be more robust, to operate in real-time, and tackle challenging tracking and scene reconstruction tasks.

In another work[24], authors approach with Automatic Model Selection, a state-of-the-art hybrid approach for object detection performance improvement is introduced by leveraging remote sensing. This hybrid method combines the best features inherited from the YOLOv7, which shows exceptional local object detection capabilities, and the Detection Transformer proficient in capturing global context. By incorporating the two models, the precision in the localization of objects becomes increasingly improved. It includes a separate module for automatic selection between YOLOv7 and DETR based on performance to intelligently select the better one in various scenes, optimize the detection accuracy in each, and improve it. Through empirical experiments, the authors verify the better performance of the hybrid model that it runs ahead of the individual detections constantly. The hitherto inherent difficulties of object detection in remote sensing images with large scales and complexities have been tackled to improve this hybrid approach on accuracy-computational efficiency trade-offs. This innovation structurally qualifies the method to be employed in real-time applications for geospatial information extraction and other critical remote sensing tasks.

The paper[25] by Khan, Lee, and Lim introduced a hybrid model for object detection to enhance the accuracy and speed of detection in self-driving cars. The work has

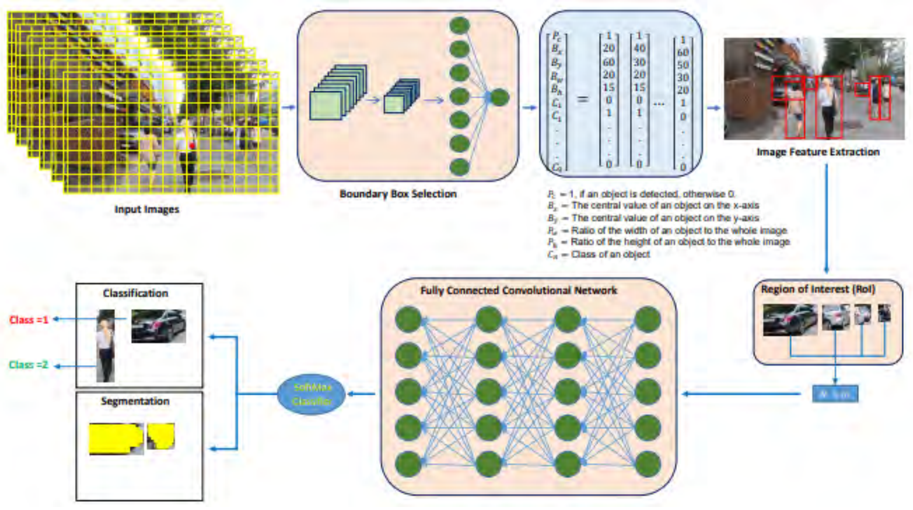

Input Images
Boundary Box Selection
Image Feature Extraction
P_c = 1, if an object is detected, otherwise 0.
B_x = The central value of an object on the x-axis
B_y = The central value of an object on the y-axis
P_w = Ratio of the width of an object to the whole image
P_h = Ratio of the height of an object to the whole image
C_n = Class of an object
Region of Interest (RoI)
Fully Connected Convolutional Network
SoftMax Classifier
Classification
Class =1
Class =2
Segmentation

Figure 2.7: Hybrid object detection model for real-time object detection in autonomous driving scenarios[25]

used the best features of YOLO and Faster R-CNN architectures in constructing an effective and efficient model. The hybrid model now skips the processing of the RPN, saving a lot of time without losing much accuracy. The authors tested their model against 10,000 labeled images of traffic objects, showing thereby that the model outperforms YOLOv5 and YOLOv7 in accuracy and maintains real-world realistic processing speeds. This good combination of real-time processing capability and high accuracy fits the model very much for an autonomous driving environment. This approach has improved object detection, especially in tough driving conditions, without compromising its capability for different types of object detection: vehicles, pedestrians, and all sorts of traffic signs.

Chapter 3

Workplan

The detection of objects in state-of-the-art autonomous vehicles is aimed at improving efficiency by understanding the depth and classification of the object to make informed decisions. For this, we have divided our work into two teams: one for the most suitable object detection models and the other for depth estimation models. This divide allowed us to explore and evaluate numerous models simultaneously, hence assuring the selection of the best options for both tasks. Second, object detection models were developed in the group; the models proposed in our paper were chosen on grounds of speed, accuracy, and adaptability to the dataset of Bangladesh on-road scenarios. At the same time, the depth estimation team worked on models that could estimate depth from the monocular pictures with high precision while allowing the measurement of the distance that identified objects are from the camera. Depth estimation is a very useful process in applications like Autonomous Driving, for which the understanding of relative distance to various objects plays a very important role in terms of safety as well as decision making.

We selected appropriate models for object detection and depth estimation to initiate the data collection process. Most of it is road-view images, collected by ourselves, which hold excellent relevance to the road conditions in Bangladesh. Then, we labeled all images, drawing bounding boxes so the dataset would work for training detection and depth estimation models. Finally, these newly annotated images were combined with earlier-acquired public datasets into one dataset. After having identified and finalized the dataset, we then went to the pre-processing stage and preparation of data to train the model. It included preprocessing the images-heavily auto-orienting them for consistency and removing superfluous or incomplete data.

Then, we divided the dataset into three independent parts: 80% of the data for training, 15% for the validation test to improve hyperparameters, and the rest dedicated to the performance evaluation of the models. Data augmentation techniques used to increase the variety in our training data were as follows: flipping, shearing, brightness correction, blurring, and adding noise. We started with using the default hyperparameters for selected models of object detection and depth estimation. Then, by training the models, we evaluated their performance, putting much emphasis on criteria involving accuracy, precision, recall, and inference speed. Following these results, modifications were done to these hyperparameters to further optimize their performances. After being satisfied with the performance of each model, we began the evaluation process for depth estimation models.

After finalizing the proper depth estimation model for our autonomous vehicle, then we started the integration phase wherein we integrated the object detection model with the depth estimation model. Integration allowed the system to conduct object detection and depth estimation, enhancing the system's capabilities to make educated judgments based on object classification and depth within the environment. The main point of our post-processing after integration was to calculate an absolute distance from the camera of the detected objects, using the obtained depth map coming out of the integrated model. This requires the use of depth data from the estimation and amalgamates it with object bounding boxes that were identified using the detection model. Further ahead, we juxtaposed these results against known real-world distances to evaluate the accuracy of our distance estimations. This was an essential stage to test the performance of all integrated systems in practical applications of, for example, autonomous driving or object tracking where the accuracy in measurements in regard to the objects' distances determines a safety and efficiency basis of making decisions. While carrying out the above-mentioned, we continued to evaluate and refine our developed approach with the goal of setting up an object-detecting-and-depth-estimating system for most real-world contexts. This integration has brought further development towards a reliable, high-performance model suitable for real-time applications in autonomous vehicles by effectively identifying objects and estimating depth precisely.

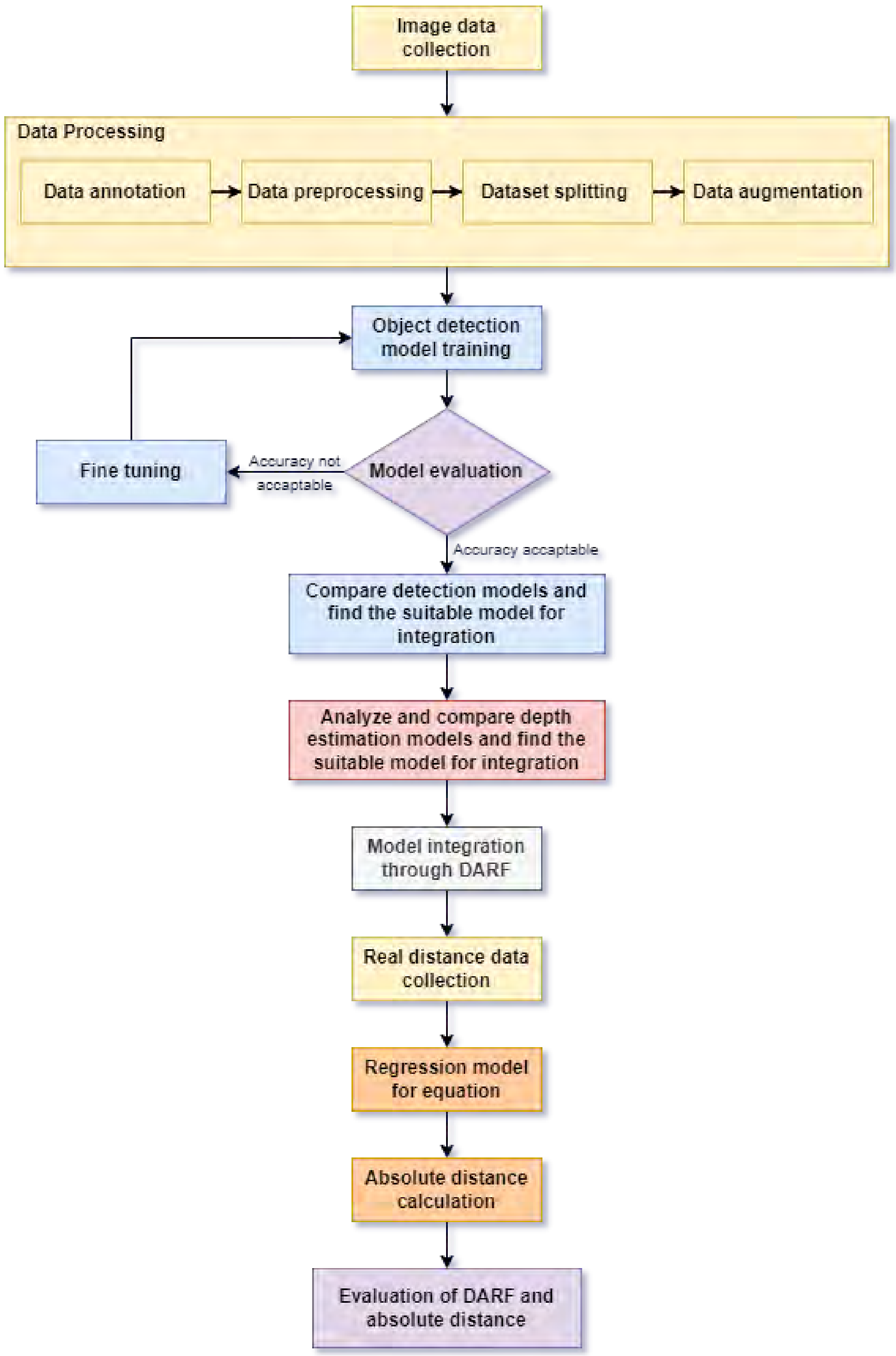


Image data collection
Data Processing
Data annotation
Data preprocessing
Dataset splitting
Data augmentation
Object detection model training
Fine tuning
Accuracy not accaptable
Model evaluation
Accuracy accaptable
Compare detection models and find the suitable model for integration
Analyze and compare depth estimation models and find the suitable model for integration
Model integration through DARF
Real distance data collection
Regression model for equation
Absolute distance calculation
Evaluation of DARF and absolute distance

Figure 3.1: Workplan flowchart

Chapter 4

Description of the Models

4.1 Object Detection Models

Object detection, sometimes also referred to as object recognition, is an important sub-segment of computer vision because it has a wide range of real-world applications based on the process of detection, recognition, and localisation. In supervised machine learning, the first category of regression problems involves classification problems as the second category. However, the quintessential classification problem is also applicable to image classification. If localisation is to be affected, it keeps categorizing an object within an image. Deep neural networks are to be used to outline the rectangular boundaries enclosing an object, known as a bounding box. Feature extraction, classification, and localisation are further steps in this object detecting problem. The object detector is identified as the two-stage object detector. The first step makes use of the region proposal network to generate RoI. While the later stage serves to predict objects and delineate bounding boxes for scenario analysis, the former stage selects credible region proposals through different strategies, including negative proposal sampling. The RCNNs, which include the Faster RCNN model, stand among the most prominent in this domain. Their advantage is that single-shot object detectors use an end-to-end streamlined architecture for the detection of a single item, including all region proposals. Taking into account all the spatial dimensions of an image simultaneously, the detectors generate class-specific probabilities for the essential objects along with bounding boxes. Two-stage object detectors outperform the single-shot object detectors by focusing only on highly probable sites for object recognition.

However, single-shot methods toward this goal have received high acclamation ever since the introduction of YOLO and all its variants. In these approaches, deep neural networks are used to cast the localization problem as a regression one. Other than YOLO, many approaches make use of SSD for object detection.

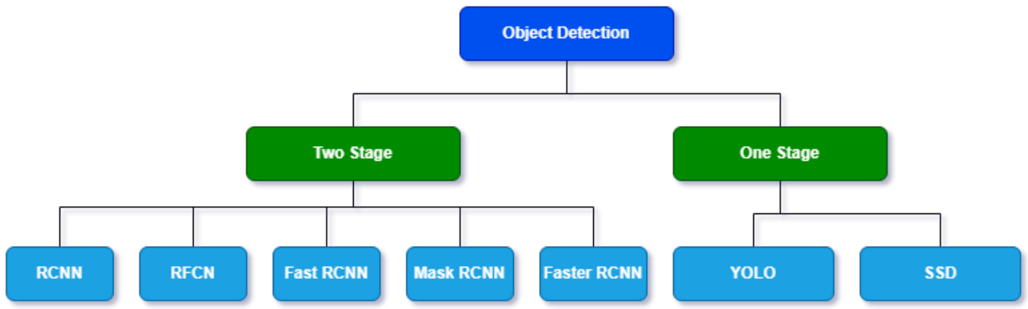


Figure 4.1: Types of object detection models

4.1.1 Faster RCNN

Faster R-CNN is currently the most common object detection model, improving upon two earlier versions of R-CNN by adding both speed and accuracy. It works in two steps: first, a proposed Region Proposal Network generates a set of proposed areas called proposal regions where objects can exist within a scene. These regions are then fed to a second stage where a CNN extracts features from each proposal. It then categorizes the objects in those regions and refines the bounding boxes to precisely locate the objects. Among the major benefits of Faster R-CNN is its accuracy. Using a separate RPN for generating region proposals avoids computational inefficiencies inherent in older models that relied on external methods, such as selective search. By incorporating this RPN into an overall network, it allows for end-to-end training. This gives the ability to train the whole model together for better performance. Therefore, Faster R-CNN is more suitable in a situation where high precision is needed, such as medical imaging and security systems.

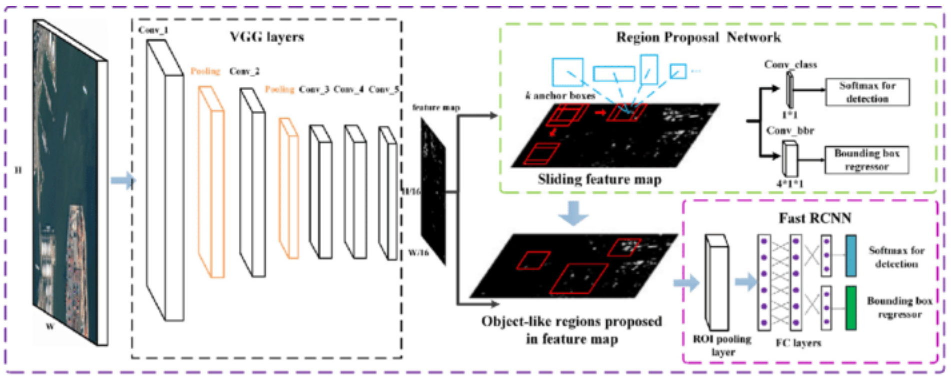


Figure 4.2: Faster RCNN architecture

Despite this, Faster R-CNN has several major drawbacks. First and foremost, its speed. While faster than previous versions of R-CNN, it is still slower compared to one-stage detectors like YOLO and SSD on real-time performance. Faster R-CNN requires a proposal region stage before classification, making it less suitable for an application that requires real-time detection such as a navigation drone or live surveillance system[12]. Another problem is the computational complexity. Faster R-CNN encompasses a two-stage architecture that, in turn, raises the computation burden and therefore becomes less efficient on various devices with constrained processing capabilities, which could envelope highly mobile platforms or embedded systems. The model also struggles to detect smaller objects or objects with unusual aspect ratios on account of limitations within its region proposal method.

4.1.2 YOLOv5

Ultralytics developed YOLOv5 as a family of object detection models for real-time applications, building upon the success of its predecessors. While not conforming precisely to the naming convention set by YOLOv4, YOLOv5 has grown into a robust, wide model in many applications. It retains the basic principle of YOLO which is a single neural network predicts bounding boxes and class probabilities over the entirety of an image and implements other architectural improvements in the quest for further gains in speed and accuracy. The architecture of YOLOv5 used CSP-Darknet53 as the backbone, which is a modification of the original Darknet design by incorporating Cross Stage Partial Networks (CSP). The design helps improve the gradient flow and reduces computing complexity, hence improving efficiency. The neck combines various segments of the feature maps from the backbone, integrating PANet for the better flow of spatial information. Detection head predicts bounding boxes, objectness scores, and class labels by using an anchor-based methodology modifying bounding boxes around pre-computed anchors. YOLOv5 works on different scales, which besides everything else makes it capable of detecting objects of different sizes in a single run[29].

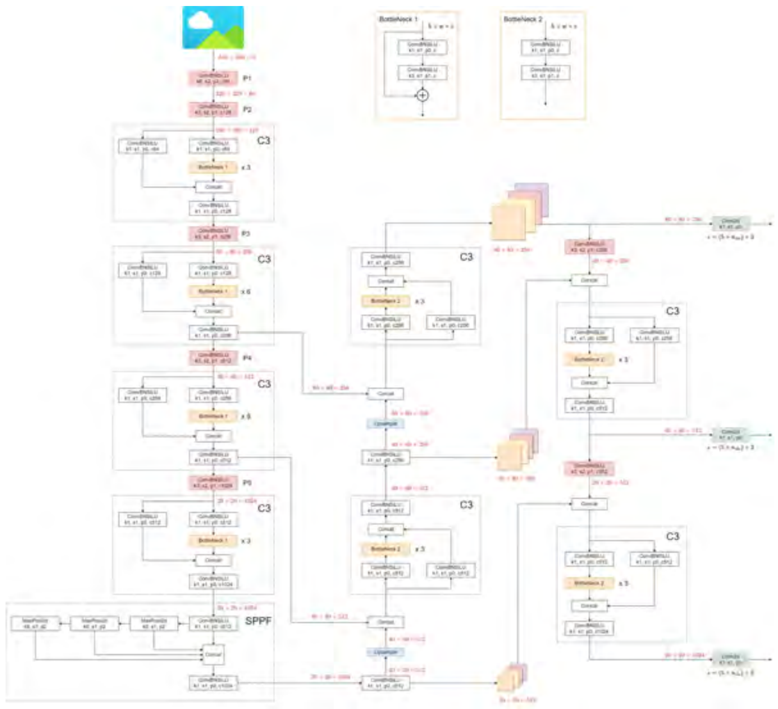


Figure 4.3: YOLOv5 architecture

The big plus for YOLOv5 is that it is fast and optimized for real-time object detection; it runs either on GPU or CPU machines for great deployment flexibility. Its versatile model allows for the effective fine-tuning of transfer learning on specialized datasets. This easily allows adaptation for new tasks after pre-training on extensive datasets such as COCO. Further, YOLOv5 was integrated with PyTorch, making it more approachable and user-friendly compared to previous versions, which were built on Darknet. It is able to perform competitive accuracy in benchmarks like COCO, speed, and performance. It also provided many model sizes: n, s, m, l, x, which, if needed, allowed the user to choose between speed and accuracy. Nevertheless, several shortcomings need consideration. Other critics also argue that YOLOv5 does not bring significant improvements over YOLOv4. Some of them do not even consider YOLOv5 to be in the same family because it was developed by a completely different team[23]. The use of anchor boxes, while convenient, is another issue that requires very careful tuning. This may be a limitation because object sizes are too diverse in data type and require more labor in tuning anchor boxes for the best performance. The use of YOLOv5 is also somewhat limited because of its GPL-3.0 licensing, particularly in the commercial sphere.

4.1.3 YOLOv8

YOLOv8 is actually the most powerful object detection model in the YOLO family, which was created on the basis of real-time detection and improved performance compared to its predecessors. YOLOv8 incorporates all basic ideas from previous versions and enhances their efficiency in terms of both accuracy and efficiency. The backbone performs feature extraction from input images. Simultaneously, the backbone of YOLOv8 is a very efficient feature extractor, like CSPDarknet53; it grants this model the ability to catch detailed information over many scales while keeping the computational costs minimal[29]. These features are integrated and fused in the neck to the various scales using the Path Aggregation Network in YOLOv8. This will allow the detection of different-sized objects, an essential attribute with big and small objects present in one image, such as trucks and pedestrians, respectively. The head in YOLOv8 predicts bounding boxes, object classes, and confidence scores. Anchor-based bounding box regression makes predictions over multiple layers for high detection accuracy. The most important advantage of YOLOv8 is a balance between speed and accuracy; because of its optimized convolution operation and model pruning methods, it's highly efficient. This enables real-time detection even on devices with very limited processing power, such as even mobile phones or drones. Most importantly, YOLOv8 was improved from previous developed versions by having better recognition accuracy in small objects due to an improved backbone network and enhanced multi-scale feature fusion. Various data augmentation and anchor generation methods improve the generalization capability of the model across many contexts.

The major advantages of YOLOv8 are real-time detection, can handle different object sizes in one image, and are easy to approach[27]. And it supports popular machine learning frameworks like PyTorch and is thus very flexible for any developer. Moreover, the streamlined architecture of YOLOv8 and compatibility with model compression facilitate its deployment on edge devices with limited performance degradation. Be that as it may, there is always a trade-off. Similar to other single-stage object detectors, YOLOv8 might fail when there are highly overlap-

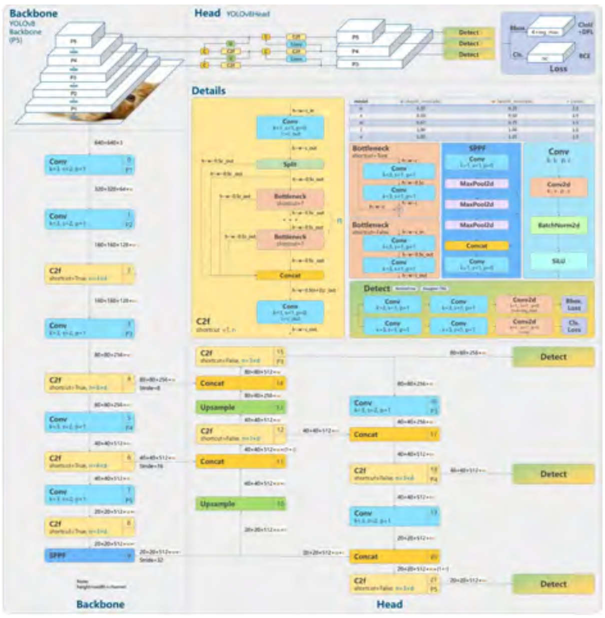


Figure 4.4: YOLOv8 architecture

ping objects. While it has enhanced small-item detection compared to predecessor models, still one can expect suboptimal performance on extremely small things in complex situations. Also, this is an internal trade-off between speed and precision, which has to be compromised to get the best accuracy in computing efficiency. For applications that strictly require the highest accuracy, YOLOv8 may not be the best compared to two-stage detectors like Faster R-CNN. YOLOv8 will be a very good choice for real-time applications that need both swiftness and high accuracy, such as autonomous driving and oversight systems.

4.1.4 YOLOv9

YOLOv9 is a leap in peer-to-peer object detection, going by the release in February 2024. YOLOv9 has improved earlier versions with an inclusion of updates that speed up and make more accurate computations with least cost in computational expense. The first two important architectural components in YOLOv9 are Programmable Gradient Information and the Generalised Efficient Layer Aggregation Network. Programmable Gradient Information The information bottleneck that sometimes arises with deep networks is partially mitigated by PGI. It guarantees stable gradient transmission for the updates to be more precise during training and becomes vital for both lightweight and deep models. This network architecture contains an auxiliary branch that retains the critical information throughout the network, hence enabling correct computation of gradients without any extra complexity at the time of inference. GELAN elevates the model's capability of effective feature collection by fusing the merits of CSPNet and ELAN, two earlier network architectures. This architecture improves the utilization of parameters and allows YOLOv9 to achieve much better accuracy on much lower computational resources compared to the previous YOLO models[30]. GELAN is especially useful for fast inference and makes sure YOLOv9 keeps real-time detection capabilities.

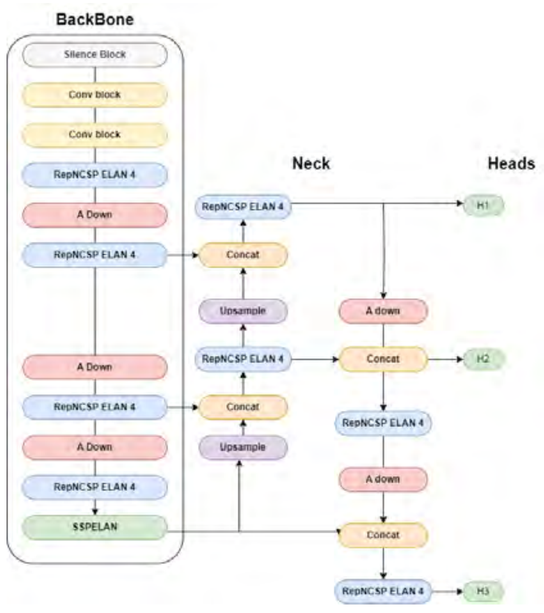


Figure 4.5: YOLOv9 architecture

YOLOv9 outperforms its predecessors by about 10% fewer parameters and 5-15% fewer calculations and improves AP by 0.4-0.6%, which makes it very efficient for both lightweight and larger models. The variant model configurations are available for YOLOv9-S, YOLOv9-M, YOLOv9-C, and YOLOv9-E to suit different applications in terms of speed versus accuracy trade-offs.

4.1.5 YOLOv11

YOLOv11 is a new generation in the series of YOLO, which was introduced by Ultralytics in September 2024, taking real-time object detection to the next level. The model comes with certain key upgrades compared to previous versions, with focus on improvements in speed, accuracy, and computational efficiency. The most striking feature of YOLOv11 is that it comes with a more robust architecture of feature extraction, enabling the detection of even more complex patterns in images when images have poor light or very congested backdrops. This uses the generalized efficient layer aggregation network as an improved form of earlier architectures to improve the utilization of the parameters, hence its improved accuracy while keeping computational resources minimal. Lighter-weight and more appropriate for edge devices, YOLOv11 retains excellent performance for real-time applications. PGI is an invention of a very fundamental level that solves the information degradation problem of deep layers in neural networks. PGI ensures that information critical to this network will vanish during training, hence improving its performance on those entangled tasks such as object detection, instance segmentation, and keypoint detection for YOLOv11.

The new version, YOLOv11, established an extraordinary performance baseline on benchmarks such as the COCO dataset. For instance, YOLOv11m achieves higher mAP but uses up to 22% fewer parameters compared to YOLOv8m, hence offering

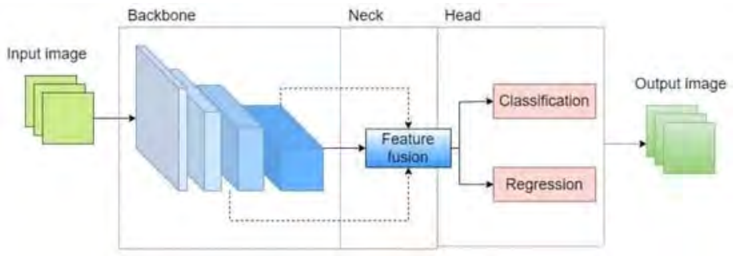


Figure 4.6: YOLOv11 architecture

better efficiency in computation without sacrificing accuracy for performance[29]. Other tasks related to object detection, such as pose estimation and oriented object detection, fall under this model. Hence, its applications are quite diversified, ranging from surveillance and health imaging to autonomous driving. YOLOv11 follows the tradition of YOLO in optimization for real-time performance. It provides fast inference performance compared to YOLOv10, and it represents a very good choice for every application where timing constraints are strict in various industries.

4.2 Monocular Depth Estimation Models

Monocular depth estimation is the process of estimating depth information from a single image without requiring multiple viewpoints or any other sensor input such as stereo cameras or LiDAR. Admittedly, this problem is fundamentally challenging due to the intrinsic ambiguity of depth cues from a single image; still, many approaches have been put forward.

Fully Supervised Models: These models require ground-truth depth map labels in their training datasets. They learn to predict the depth by minimizing the difference between the estimated and actual depth. While accurate indeed, their process is heavily dependent on huge and labeled data, which again is costly and challenging to acquire.

Self-supervised Models: They don't require explicit depth annotation supervision. They infer depth by checking on the consistency of views in an image sequence such as video frames or a stereo image pair. They use reprojection methods or disparity estimation methods. While these models already demonstrate certain tendencies toward a reduction of dependency on labeled data, their accuracy could possibly get better, and besides, they are not helpful with dynamic objects either.

Unsupervised Models: These kinds of models negate the need for depth labels by forming knowledge through unlabelled images, using indirect indicators such as optical flow or visual consistency. They can afford the best flexibility regarding the demands of data. Their performance may turn out to be even better when juxtaposed with that of the supervised models, particularly in complex or dynamic settings.

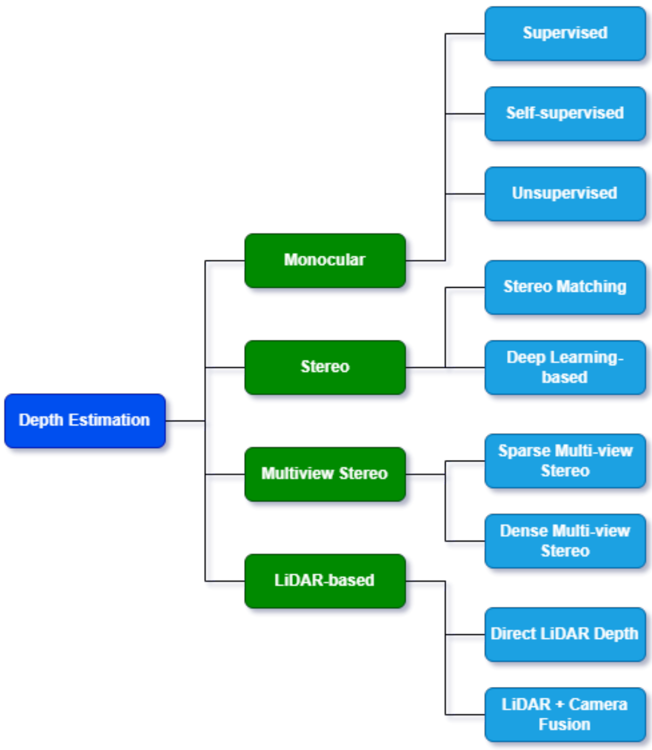


Figure 4.7: Types of depth estimation models

4.2.1 MiDaS

MiDaS is a state-of-the-art deep learning model that can predict depth from a single RGB image. It has been very successful because it generalizes well across a wide range of environments, from indoor to outdoor scenes. Generalization in this context is made possible by training on large and diverse datasets such as ReDWeb, MegaDepth, and ETH3D[15][10][33], which expose it to a wide variety of scenes, lighting, and camera viewpoints. The architecture used for MiDaS is based on an FPN backbone; hence, it is able to allow multi-scale extraction of depth features. It also helps in capturing finer local details along with greater global structure, hence enabling the model to infer both relative and absolute depth. All these characteristics make MiDaS highly applicable in autonomous driving, robotics, and even AR applications, since it is highly feasible for obtaining estimation in decision-making over dynamic environments easily[19].

In figure 4.8, On the left architecture overview. The input image is first tokenized, either by extracting non-overlapping patches which is followed by a linear projection of its flattened representation, or by applying a ResNet-50 feature extractor. These can be seen in DPT-Base and DPT-Large, and DPT-Hybrid, respectively. A positional embedding adds to the image embedding, while a patch-independent readout token-in red-is added. Tokens go through many transformer stages. We reassemble tokens from different stages into an image-like representation at multiple resolutions (green). Fusion modules (purple) progressively fuse and upsample the representations to yield a fine-grained prediction. In the center, the overview of the Reassembles operation. Tokens are assembled to feature maps with 1/s the spatial

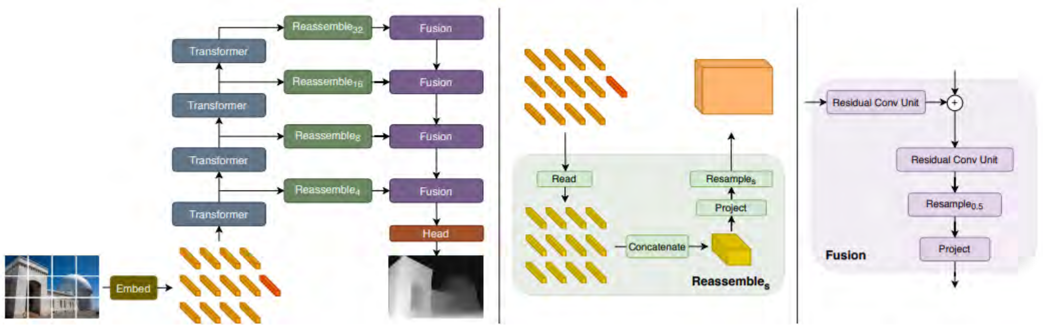


Figure 4.8: MiDaS architecture[21]

resolution of the input image. On the right, fusion blocks combine features using residual convolutional units and upsample the feature maps[21].

MiDaS_small is designed to be much lighter and targeted for real-time depth estimation on resource-limited conditions compared with MiDaS v2.1. Powered with an EfficientNet-Lite3 backbone, MiDaS_small is now able to process images up to a resolution of 256x256; thus, significantly more inference speed, with reasonable accuracy maintained. In this work, we propose the MiDaS_small model, which is aimed at running on mobile and embedded systems such as drones, mobile robotics, and augmented reality devices, where speed and efficiency are extremely needed. This lightweight model provides good opportunities for performing realistic real-time depth estimation under resource-constrained environments by harmoniously balancing performance with computational cost, although it slightly sacrifices some precision to its bigger siblings.

MiDaS v3 introduces the next generation to the depth estimation model, further integrating transformer-based architectures into the model architecture for even more accuracy and robustness. The two most important models in MiDaS v3 are DPT_large and DPT_hybrid. First is the model DPT_large, which is a fully transformer-based approach comprising only vision transformers that participate in extracting global depth information from images. That is, because of this, the DPT_large architecture is able to model complex spatial relationships involving complete scenes; in this respect, it is highly accurate in depth prediction. The former is exemplary for precision-bound tasks like 3D reconstruction and detailed environmental mapping because of typical use, but at higher computational intensity, is ill-suited for real-time applications.

On the contrary, DPT_hybrid balances speed and accuracy by reaping the relative benefits of CNNs and vision transformers. For DPT_hybrid, the local features are extracted with CNNs. The long-range dependencies and global context of the scene are modeled by the vision transformers. Working at 384x384 resolution, DPT_hybrid maintains higher accuracy compared to smaller models like MiDaS_small, at a degree of computational efficiency suitable for real-time applications[21]. It is especially suitable for such tasks as autonomous driving and real-time robotics, where it deals with performance and precision.

4.2.2 MonoDepth2

MonoDepth2 predicts the depth map from a single RGB image input given. It is a self-supervised monocular depth estimation model that requires no ground-truth depth information in training and, on the other hand, depends on photometric consistency among the input views. The specific model will jointly learn depth and camera pose, either from stereo image pairs or from monocular video sequences. MonoDepth2 relies on view synthesis during training, where the model predicts the appearance of one image from another viewpoint given estimated depth and camera pose; this further compares the synthesized view with the actual image for minimizing photometric error and hence allows the model to indirectly learn about depth[9]. It combines photometric reconstruction loss, depth smoothness loss, and per-pixel minimum reprojection loss to handle occlusions and provide more depth map accuracy itself[3].

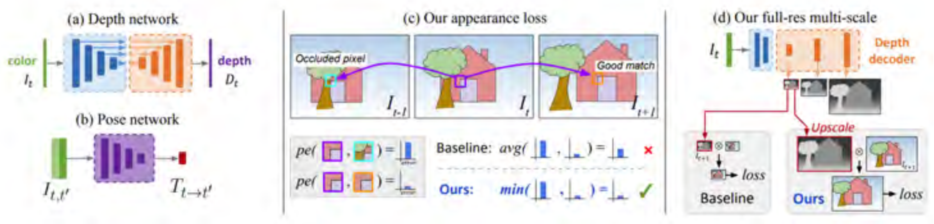


Figure 4.9: MonoDepth2 architecture[32]

Architecturally, the MonoDepth2 architecture follows an encoder-decoder structure commonly used in many computer vision tasks. It leverages a ResNet-based encoder structure, either ResNet18 or ResNet50, to extract high-level features from the input imagery[2]. These features are then upsampled by the decoder into a full-resolution depth map. MonoDepth2 innovates with its multi-scale prediction of the depth, thereby progressively refining the depth estimates across different scales, resulting in more accurate and detailed depth maps. It also predicts the relative camera motion between frames apart from a depth network with the help of the pose network, hence the model works even when only monocular video sequences are provided to the model.

MonoDepth2 was trained using stereo image pairs and monocular video sequences, both of which contain KITTI, a dataset of road driving data captured from vehicle-mounted cameras. Stereo training mode utilizes images from both the left and the right within a stereo pair for providing supervision in the estimation of depth. Monocular training mode, on the other hand, assumes that there may be camera motion between successive frames, thus making use of such to make predictions about depth. Besides self-supervision, MonoDepth2 provides very accurate depth maps and is as efficient for real-time applications; therefore, this model finds a wide application in the tasks related to autonomous driving, robotics, and augmented reality.

4.2.3 MegaDepth

MegaDepth is a deep learning-based depth estimation model that generates high-quality depth maps from single RGB images. Unlike self-supervised models, such as MonoDepth2, which rely on large datasets of MVS images acquired from internet photos, this primarily focuses on large-scale outdoor and urban environments. It is designed to handle a wide range of general settings, including complex outdoor scenes, making it suitable for applications like 3D reconstruction and Virtual Reality. MegaDepth builds detailed depth representations by matching multiple views of the same scene through multi view stereo images and is therefore trained as such.

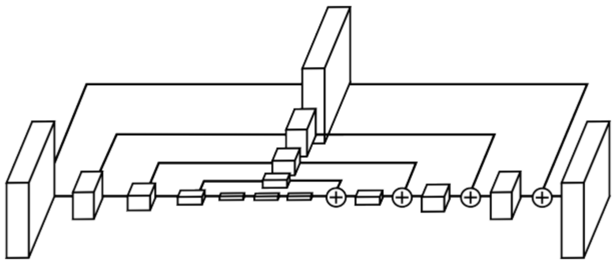


Figure 4.10: MegaDepth (hourglass) architecture[5]

Its strong point is that it shows good performance for the uncontrolled setting using internet images with quite diverse lighting conditions, occlusions, and viewpoints. It leverages ground truth depth maps created from multiview stereo techniques at training time to perform much better at the inference time when it has to estimate high-resolution depth given a single image[11]. In other words, the network has an encoder-decoder structure: it uses an encoder to extract features of an image via a deep residual network and utilizes those extracted features through the decoder by upsampling in order to predict a dense depth map. Multi-scale depth prediction further enhances the accuracy of the model for each of their different spatial resolutions.

Another salient feature for MegaDepth is that there is generalization across various scene types, based on a large and varied training dataset with inclusion of diverse internet images collected across an array of geographically dispersed locations insofar as weather conditions are concerned. It consists of over 1 million images, together with the estimated depth from the structure-from-motion and multi-view stereo estimates that might as well train MegaDepth. This diverse data allows MegaDepth to perform especially well on outdoor scenes featuring different visual structures, often found around famous landmarks and urban landscapes[6].

While MegaDepth is well-performing for depth prediction, both in an outdoor and large-scale environment setting, it can be computationally expensive since the model size is bigger, and it was computed with the help of a high-resolution depth map. Its strengths make it very suitable for applications such as 3D reconstruction, augmented reality, and virtual reality[7].

Chapter 5

Description of the Dataset

Data is an integral part of all AI models and, in many ways, the only reason machine learning is becoming so popular right now. Because more data is available, scalable machine learning methods can now be used as complete solutions to improve autonomous vehicle projects. We faced significant challenges in making our dataset due to the lack of suitable Road View datasets that met our needs. However, we persevered and put together a collection of around ten thousand images. This dataset allows us to analyze our integrated model properly.

5.1 Process of Data Collection

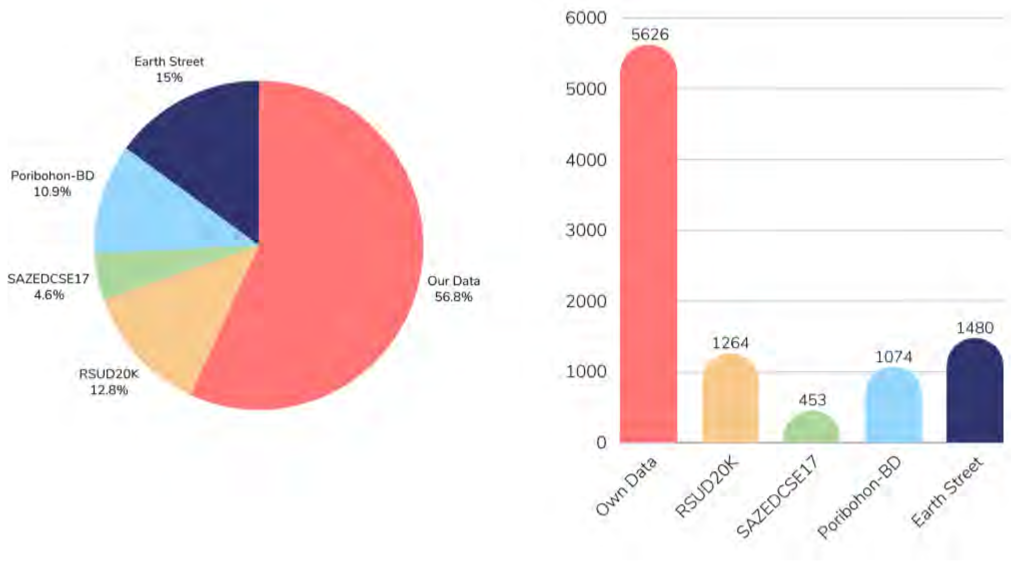


Figure 5.1: Dataset source distribution

The data collection process was time-consuming and extensive. While there are numerous publicly accessible road view datasets, most of them are based on foreign road scenes that are substantially different from those in Bangladesh. As this paper focuses on the Bangladeshi road environment, which is unique with its human haulers, auto rickshaws, and rickshaws, we had to be extremely cautious when considering the use of existing datasets. This caution was necessary to ensure the quality and relevance of our dataset.

Fortunately, we were able to locate several pertinent datasets that were explicitly designed for Bangladeshi roads. Nevertheless, these datasets needed to be revised in the specific vehicles and objects necessary for our endeavor. Also, these existing datasets were not balanced as required by the models. Consequently, we were compelled to exercise caution in utilizing pre-existing data, augmenting it with our data collection endeavors. Our dataset comprises 9,897 images (28000 bounding boxes approx.), of which we acquired 56.8%. The remaining images were obtained from the SAZEDCSE17[26], Poribohon-BD[18], and RSUD20K[31] datasets. Also, we collected images from Earth Street[28] videos to diversify the dataset further.

Furthermore, we guarantee that our dataset encompasses a broad spectrum of weather conditions, including harsh sunlight, rainy, overcast, and night time scenarios, to train our models to function efficiently in various environments. The graph below depicts the distribution of images acquired from various sources and the range of weather conditions represented in the dataset.

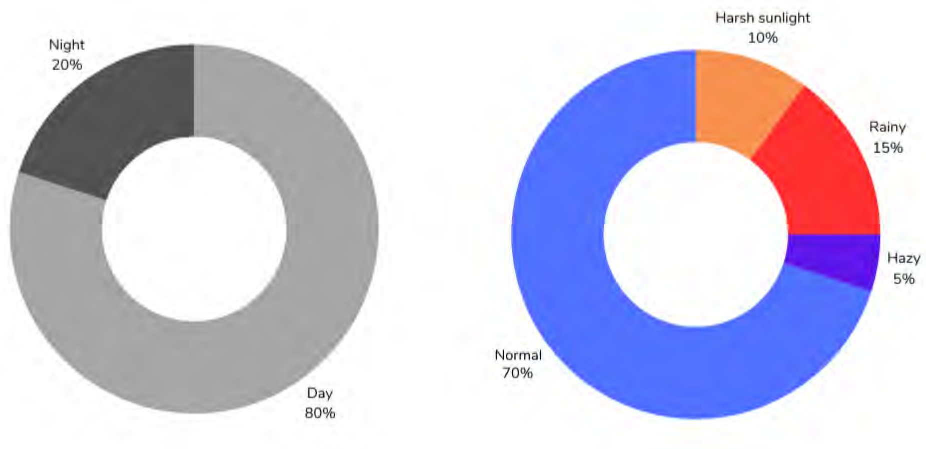


Figure 5.2: Dataset distribution on weather basis

5.2 Hassle of Data Collection

Going across Bangladesh extensively, this ensures that the dataset should be representative and from Bangladesh perspective, in trying to create a dataset that would best represent road sceneries of truly Bangladeshi origin. We collected pictures from several locations including, but not limited to, Dhaka, Chattogram, Dinajpur, Rangpur, and Bogura, which capture well a wide range of road vistas and surroundings.

However, certain difficulties were found in collecting the dataset. The diversity in vehicle types across regions added a peculiar challenge. For example, some neighborhoods reflected more commercial or heavy-duty vehicles, while on some roads, the traffic of rickshaws and pedestrians is disproportionately large. Such an imbalance, particularly in respect of under-represented vehicle types, has necessitated focused efforts at data collection just to ensure the integrity of the dataset for purposes of training an autonomous car to base its driving decisions on. This exhaustive and varied approach added much more complexity but also significantly enhanced richness and key utility in regard to autonomous driving systems in Bangladesh.

5.3 Data Annotation

They are necessary since this process is critical to the development of raw images into data that is suitable for machine learning model development. Annotation entails the labeling of images and tagging various objects in images as a way of making it easy for the model to identify and understand various things and their attributes. It promises an accurate, improved understanding of the model's ability to understand different inputs through visual means in making proper predictions.

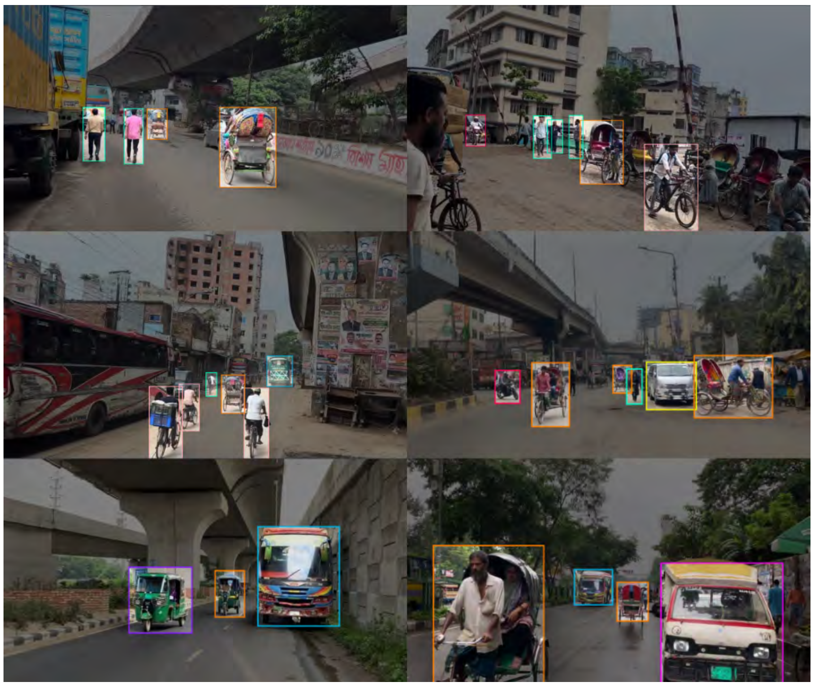


Figure 5.3: Data annotation of raw images

Moreover, Roboflow was used to effectively annotate images online in an intuitive way. Roboflow had an easy interface with a full suite of tools that streamlined the process of marking or labeling objects within the images. On this platform, we did a manual annotation of roughly 20,000 bounding boxes, ensuring that the annotations are of high quality and accurate. Besides this, our annotations included metadata such as the position of objects within bounding boxes, item categories, and other crucial information necessary to train the model effectively. In incorporating bounding boxes, the model understood and recognized objects by precisely defining the relative spatial relationship with other objects in the image. This typical annotating process was extended to our entire dataset with consistency in the annotation process using Roboflow, preparing the dataset to be fairly reliable for model training.

5.4 Labeled Classification

In our dataset, the categorized successful objects into 11 distinct classes are bike, pedestrian, car, rickshaw, bus, truck, CNG, leguna, bicycle, van, and auto rickshaw. During the creation of the dataset, one of the high priorities was trying to make sure that the classes are balanced, meaning none of the particular classes dominate. This balance was achieved by diligent annotation of each category. By the time we combined our dataset with collected datasets, the cumulative sum of 28,078 annotations was obtained (9897 images). The annotations are evenly distributed to ensure the dataset is well-balanced; this is an important aspect in real training of the model and avoiding class bias. The graph below shows the number of annotations for each class, which balances out classes of the objects.

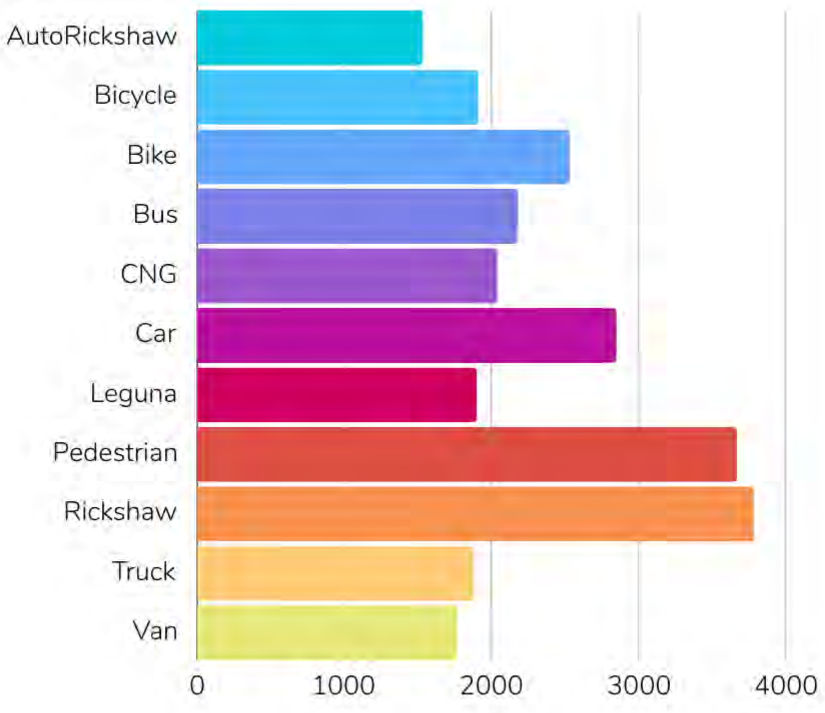


Figure 5.4: Data classification

5.5 Dataset Splitting

The data was divided into 80% for training, 15% for validation, and 5% for testing. This training set enables the model to understand many of the characteristic patterns and features inherent in it. The validation set is used to fine-tune the model.

It means that changes in the values of hyperparameters, done as per changes in performance, will be able to be tried out without the model suffering overfitting. The test set is kept for the final evaluation of the model's performance. This dataset remains untouched during training and validation phases, ensuring that the test results provide an objective measure of the model's real-world effectiveness.

5.6 Data Preprocessing

We use preprocessing to prepare the images to suit training or performing inferences on a model. That is, it prepares image data for the technical and performance-related needs of computer vision models.

5.6.1 Image Scaling

Convolutional neural networks (CNNs) require all input data to be arrays of uniform size, especially when they are fed into a fully connected layer. If there is a difference in the sizes of the images, it causes an error or makes the performance of the model poorer. Since our various images were from different sources and had different resolutions, we standardized all of them into 640x640. This ensures that all the images of the dataset are uniform in measurement, while also compressing the images to reduce file size, which helped speed up model training and made better use of computational resources.

5.6.2 Auto-orientation

Preprocessing involves auto-orientation of the images to get all the images correctly oriented. Due to the image being taken from different angles or with different camera setups, the orientations may not be consistent and may bias or confuse the model. This helped in auto-orienting every image within the dataset. The method automatically kept the orientation of images uniform, which is very important for maintaining data quality and reducing bias.

5.6.3 Filtering of Data

It filtered out some data from the dataset that contained unnecessary information. This included removing images that didn't have classes or didn't possess important data. By filtering out such data, we prepared the dataset to become cleaner and more to the point, hence serving efficient model training.

5.7 Data Augmentation

Image augmentation involves modifying images to provide diverse versions of the same data to increase the model's training exposure. It is not easy to fully encapsulate a model encompassing every imaginable real-world occurrence. This is where augmentation proves beneficial. Augmentation allows us to incorporate more scenarios that may be challenging to identify in reality. Expanding the training data enables the model to learn from various events and generalize to novel circumstances. In augmentation we applied-

5.7.1 Flip

Horizontal flipping is one of the augmentations to the dataset. This technique reflects the image across the vertical axis, producing a fresh rendition of the image in which things are depicted. Horizontal flipping enhances the model's ability to identify objects irrespective of their orientation, thus the model can find another object with a flipped vertical orientation.

5.7.2 Shear

In shearing, the horizontal or vertical axis of the picture is just slightly tilted. The horizontal and vertical axes might cause the picture to be sheared by $\pm 10°$. This change makes the picture seem different, as if seen from several vantage points. Shearing improves the model's generalisability, making it more applicable to real-world scenarios with objects seen from different perspectives.

5.7.3 Brightness

Changing picture brightness introduces a new degree of data variability for training. Image darkening or lightening may be achieved within the brightness adjustment range of -15% to +15%. This improvement allows the model to handle different lighting conditions, ensuring that it performs well in real-world environments with varying brightness levels.

5.7.4 Blur

Another enhancement was blurring, which was applied to the dataset with an intensity of up to 0.5px. Like motion blur or defocus, this method somewhat blurs the picture details. To make the model more resistant to small distortions, it may be trained on blurry pictures. This makes it possible to accurately recognise and classify objects even when the images aren't very crisp.

5.7.5 Noise

Introducing noise into the images causes random pixel alterations throughout the image, impacting up to 0.3% of the pixels. This improvement mimics actual flaws, like noise from the camera's sensor and outside elements like dust or rain.

After applying augmentation, we get a total of 23580 images, which consists of 80000 bounding boxes (approx.). This enabled the model to provide more variety of scenes and thus, the training of the models more precisely. Our dataset link-https://app.roboflow.com/thesis-v2-rwjrw/final-dataset-v5

Chapter 6

Model Training and Analysis

6.1 Object Detection Model Training and Analysis

6.1.1 Faster RCNN

The Faster R-CNN trained for 70 epochs, expresses several important metrics concerning poor performance. First, its precision score is low at 0.434, indicating that less than half of all the objects predicted are correct. This means that the model does poorly in getting the class right concerning the object detected, which is of prime importance in applications such as autonomous vehicles. Also, the recall score of 0.534 shows that the model misses the detection of nearly half of the real objects present in the images, which in turn further indicates the underperformance of the model concerning object detection. Missing the presence of objects in a critical environment could result in hazardous situations like traffic incidents; thus, requiring a more reliable model.

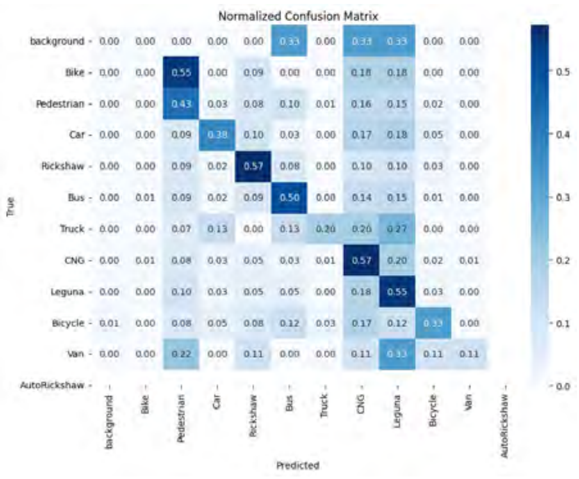


Figure 6.1: Faster RCNN confusion matrix

The F1 score is the balance of precision and recall, which was 0.594, showing quite low values and interpreting this as generally low efficiency given by the model while doing object detection. The closest possible F1 score to 1 points to a strong model, while the present one indicates that there is great room for substantial improvement. Moreover, the mAP@50 of 0.588 and mAP@50-95 of 0.401 reflect weak overall accuracy, especially with the application of more conservative thresholds of overlap IoU. This is indicative that the model generalizes poorly in cases where objects are positioned differently. This calls into question how well this model will perform in environments other than this simplified one, considering real-world environments that are complex in nature and fast-paced.

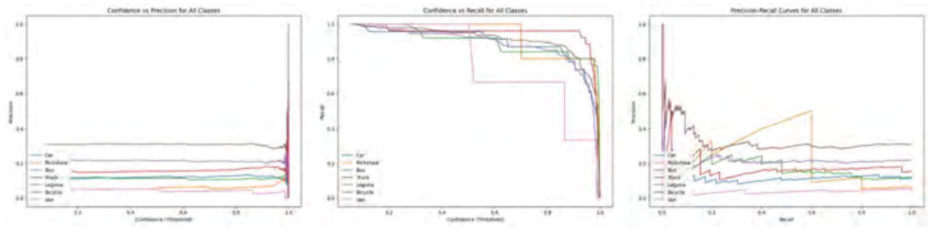


Figure 6.2: Faster RCNN Precision, Recall and Precision-Recall curve

The confusion matrix shows frequent misclassifications between similar object categories like ”Car” and ”Rickshaw”, with even poorer distinction for objects like ”Bicycle” and ”Pedestrian.” This inconsistency in the accuracy of classification also goes to support a model that is pretty far from being able to carry out complicated object detection. Finally, the graphs of precision-recall and confidence are not consistent across object classes. Although some classes perform slightly better than others, most classes, such as ”Van” and ”Bicycle,” cases show extremely poor precision and recall. Even with a rise in confidence, there is no significant improvement of either precision or recall by the model. This again highlights its limitations.

6.1.2 YOLOv5

YOLOv5 was trained on the dataset with 11 classes of people, cars, rickshaws, and other vehicles that one normally sees on roads. In this example, training was set to run for 200 epochs. However, it stopped at 173 epochs because of an early stopping condition set to patience 10. Other important training parameters are a batch size of 8 and picture dimensions of 640x640, which is pretty standard for most object detection tasks. We also used the previously trained and saved weights in yolov5s.pt. This weight was a good initial weight for object detection. However, we implemented different key hyperparameters to ensure stability in this model and improve performance during training. The learning rate was changed, with the first epoch starting from 0.0033, increasing to 0.0098 in the fifth epoch until attaining a state of stabilization. This warm-up technique of the learning rate helped in warming up the gradient descent process and reducing large initial updates that might mess with training.

Also, the training losses-box loss, classification loss, and distributional focal loss indicate that each is steadily going down during training. The box loss decreased-

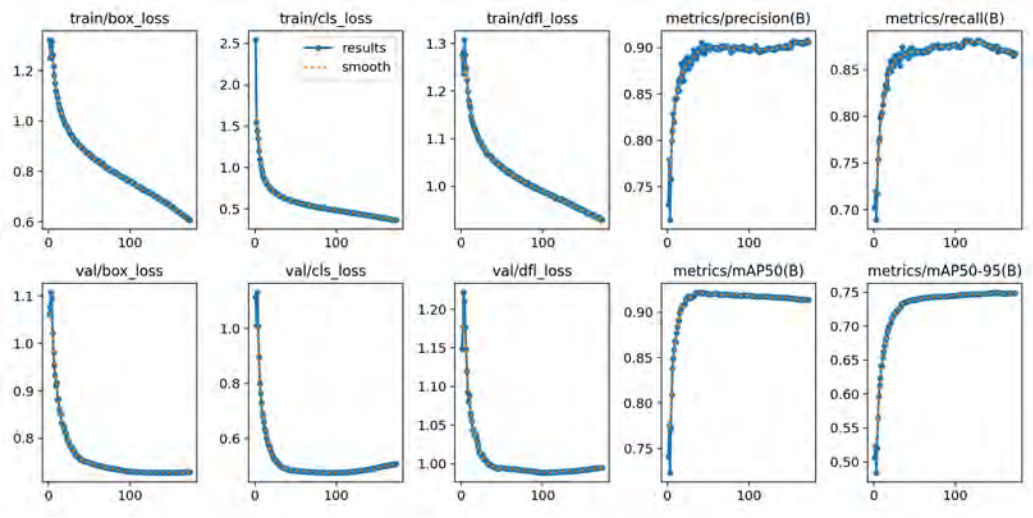


Figure 6.3: YOLOv5 Train-loss, mAP, precision and recall metrics

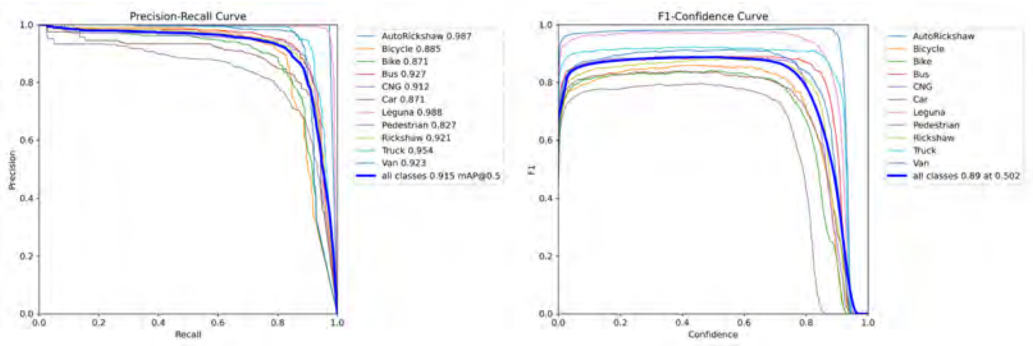


Figure 6.4: YOLOv5 Precision-Recall and f1 Curve

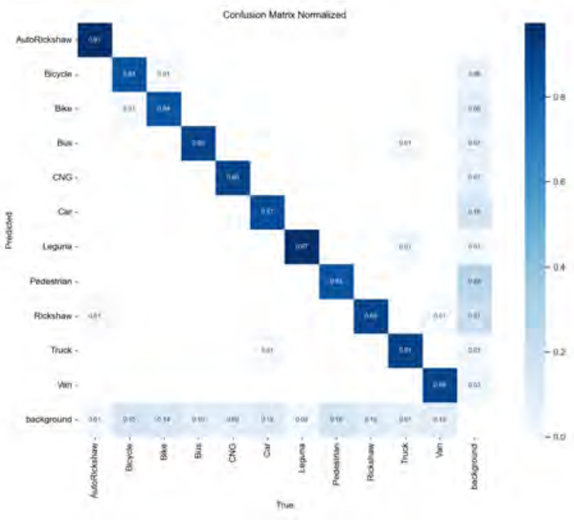


Figure 6.5: YOLOv5 confusion matrix

from about 1.32 to below 0.6-which is responsible for object localization, showing a dramatic increase in the bounds of the rectangle which the model was able to predict over time. Also, there is a reduction in classification loss-from 2.54 to below 0.6-showing how much more accurate the model became in identifying any detected objects. These are trends in validation losses that are similar, which shows the model is generalizing well to new data, and overfitting remains limited. This high precision with the detection of objects shows how strong and exact this model could be in making predictions.

The precision, recall, and mAP metrics indicate overall model performance. The precision for the last epochs was over 0.90, standing at 0.908, which really signifies that the model effectively identified items with a high actual positive rate while reducing false positives. Recall rose progressively to 0.880, showing that the model was good at zealously identifying objects across many categories. The mAP@0.50 finally converged to around 0.915, reflecting the model's pretty good performance at 0.5 IoU, while for the more informative mAP50-95, recording performance from a wider IoU range, the value is 0.748. Moreover, the confusion matrix gave some complementary information referring to the performance of each class: Very high precision of 0.97 was found in the AutoRickshaw class, while for other vehicle classes, the model showed a little inconsistencies but still could provide healthy detection rates. For pedestrians, there is a slight overlap of classes, which belonged to similar categories, but the general detection was still good.

An F1 score of 0.89 at a confidence threshold of 0.502 shows strong balancing between precision and recall for each and every class. The Precision-Recall curve showed that classes like AutoRickshaw and Leguna were doing better. But classes like Pedestrian and Bicycle showed a higher improvement; maybe it is due to the complexity of their features or their similarity with other classes.

6.1.3 YOLOv8

YOLOv8 was trained using our custom dataset, though with minor modifications to the settings. The training was set at 200 epochs but was stopped after 147 because of the early stopping mechanism with a patience of 10 set. We are using a batch size of 8, and the image dimensions are 640x640 for competing computing efficiency with optimal performance in detection. We were also applying pre-trained weights from yolov8s.pt. This weight provided a very strong backbone for object detection. The learning rate was initially set to 0.0700 while warming up and gradually decreased to 0.00985 toward the end of training. Such a change in learning rate facilitated the finding of an appropriate learning speed that would avoid huge and unpredictable updates during gradient descent.

Training losses continuously improved with respect to all metrics. Box loss decreased from 1.31 initially to below 0.8, which also indicated improved object localization. The class loss went down from 1.53 to about 0.8, indicating improvement in accuracy over classes. Regarding DFL loss, a similar trend is maintained with a steady decrease that does not indicate perfect bounding box alignment. The model has

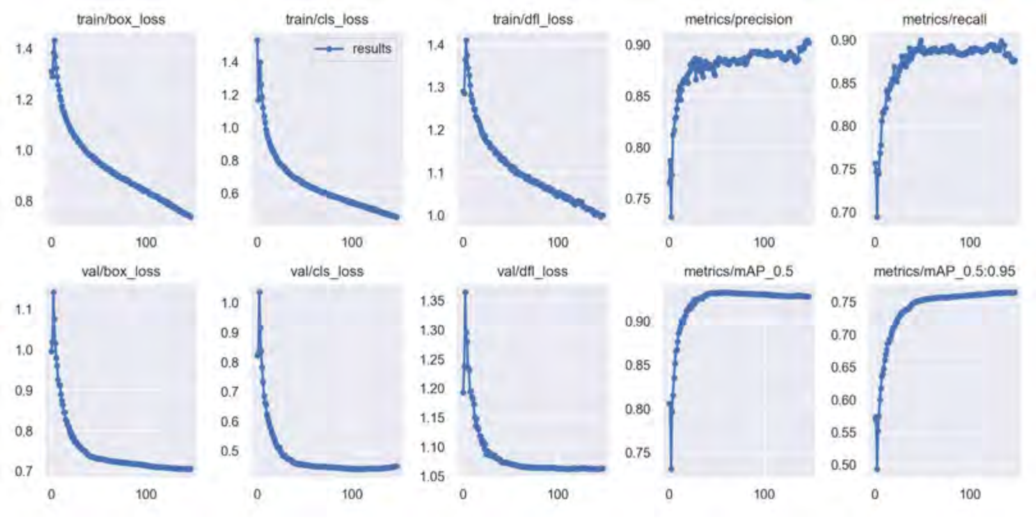


Figure 6.6: YOLOv8 Train-loss, mAP, precision and recall metrics

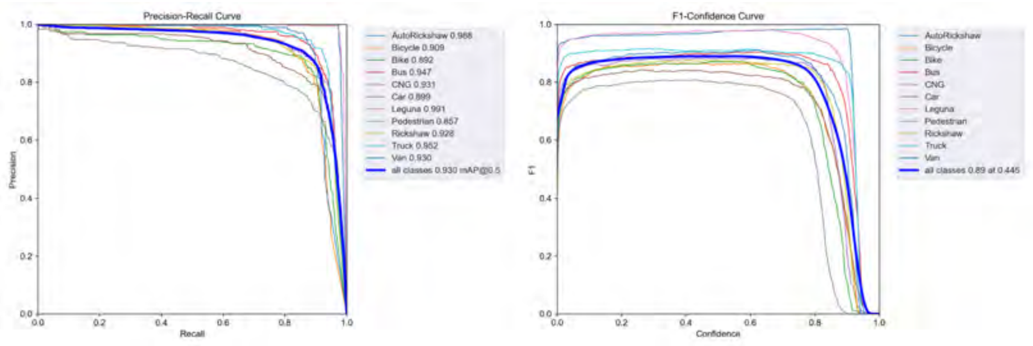


Figure 6.7: YOLOv8 Precision-Recall and f1 Curve

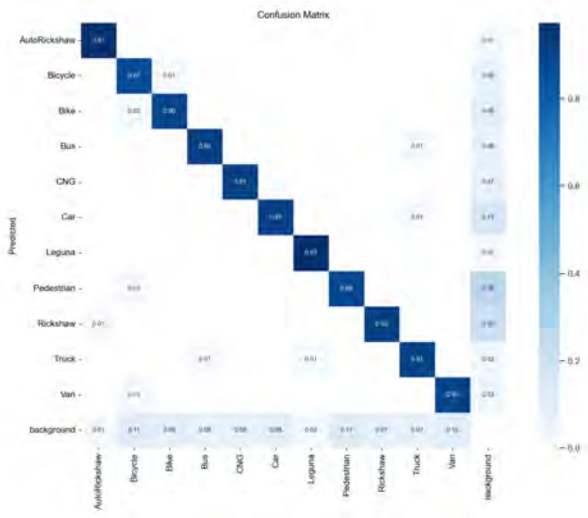


Figure 6.8: YOLOv8 confusion matrix

proved its generality by generalizing well on different classes. For validity, both box loss and class loss are decreased gradually. The loss in the validation box started from 1.0 and kept decreasing which is a good indication of the improvement in the generalization of unseen data of the model. The classification loss went down from 0.82, which indicates that among the classes, proper generalization was achieved by the model.

The precision being 0.903 depicts very good accuracy that cuts down false positives, which reassures performance regarding the model. Recall is 0.898, which shows that the model was able to detect most of the objects inside the images. The mean Average Precision at 50% Intersection over Union, mAP50, was 0.930, whereas for the mean Average Precision with IoU threshold averaging from 0.5 to 0.95, referred to as mAP50-95, was 0.764. This hints at the proficiency of the model in object detection because the objects cause different extents of occlusion.

The confusion matrix was enlightening in the thorough understanding of the performance of YOLOv8 on different classes. The class AutoRickshaw had a very high precision of 0.97 and was rarely confused with other classes. On the other hand, the Pedestrian and Bicycle classes seemed to be highly confused with other classes, although the overall performance of the model was strong. This would mean that it performed very well on most categories of objects but may need further refinement at classes which can be intricate or overlapping.

It reached 0.89 at the confidence threshold of 0.455, which showed that there is a very effective balance between Precision and recall for most of the object classes. Also, according to the Precision-Recall curve, the performances of AutoRickshaw and Leguna were better, while the Pedestrian and Bicycle classes had marginally lower scores, indicating further optimization.

6.1.4 YOLOv9

The YOLOv9 model conducted training for 200 epochs; however, with a patience parameter of 10, resulted in termination at 127 epochs. The model was trained with a batch size of 16 and an image dimension of 640x640, achieving a compromise between computational efficiency and data processing. These setups enabled the model to process more data every iteration, improving training speed while preserving memory efficiency. Further, we used the pretrained weights from yolov9s.pt. This weight provided a strong baseline for object detection. Throughout the training, the learning rate began at 0.0033 and progressively escalated to 0.00985 over the initial epochs, facilitating gradual learning and preventing substantial, destabilizing updates. During the training process, the learning rate was reduced to promote convergence.

The model's performance was monitored using various loss functions, such as box loss, class loss, and Distributional Focal Loss (DFL). The box loss, which quantifies object localisation, consistently declined from 1.32 to approximately 0.8, signifying enhanced bounding box precision. The class loss demonstrated significant enhance-

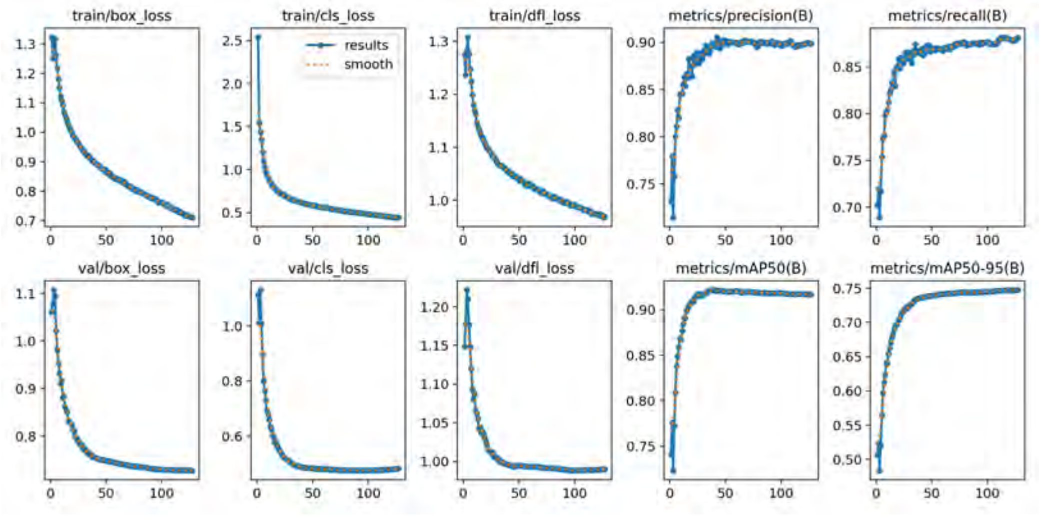


Figure 6.9: YOLOv9 Train-loss, mAP, precision and recall metrics

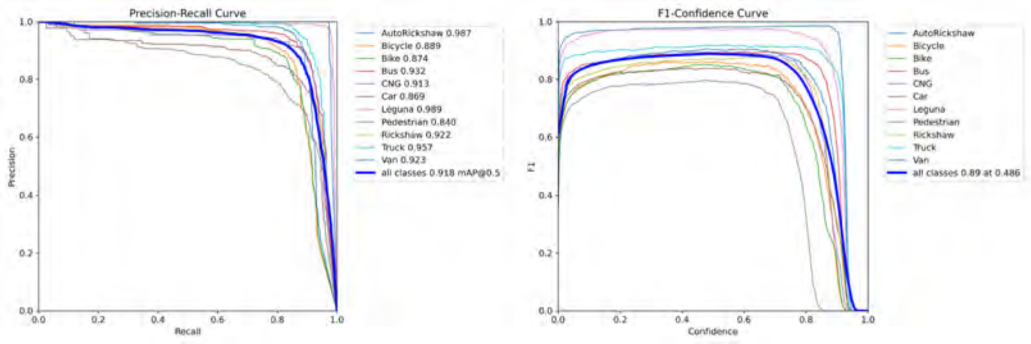


Figure 6.10: YOLOv9 Precision-Recall and f1 Curve

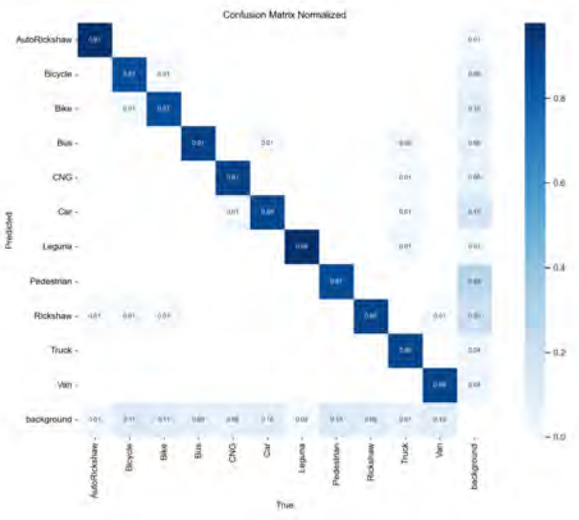


Figure 6.11: YOLOv9 confusion matrix

ment, commencing at 2.54 and declining below 1.0, indicating improved object categorisation. The DFL loss improved as the model enhanced its ability to refine bounding box coordinates.

The validation losses exhibited steady enhancement, with the validation box loss declining from around 1.06 and the validation classification loss reducing from 1.11 to 0.89, indicating the model's proficiency in generalizing to unknown data. The model's precision stabilized at approximately 0.902, signifying effective minimisation of false positives, while recall enhanced to 0.881, demonstrating robust item detecting capabilities. The mean average precision (mAP) at 50% IoU attained a value of 0.918, while the mAP across a broader spectrum of IoU thresholds (mAP50-95) achieved 0.747, underscoring the model's resilience in identifying items with differing degrees of overlap.

The confusion matrix offered a comprehensive analysis of performance across classes. Classes such as AutoRickshaw attained great accuracy with low confusion. However, classes like Bicycle and Pedestrian had some overlap with other classes, suggesting possible areas for enhancement. The F1 score, which balances precision and recall, reached a value of 0.89 at a confidence threshold of 0.486, and the Precision-Recall curve exhibited robust performance across most classes, especially AutoRickshaw and Leguna. Nevertheless, the model exhibited some difficulty with categories such as Pedestrian and Bicycle, indicating potential for additional improvement.

6.1.5 YOLOv11

YOLOv11 was trained for 200 epochs where, with a patience of 10, the training terminated after 103 epochs. We have also used the weights from yolov11s.pt which were previously trained. This weight was a quite good starting point for detecting objects. The batch size used was 16 and the image dimensions were 640 x 640. This is one configuration, in order to optimize efficiency in training while guaranteeing enough data processing per batch for optimum learning. Careful tuning of the learning rate was done throughout the process of training. Starting from 0.0033 to increasing amounts at 0.00985 in the first few epochs, actually, this heating-up warmed the model in order to avoid huge and destabilizing changes that allowed a gradual increase in its learning capacity.

There were several loss functions that showed up during training. The box loss, representing the model's skill at predicting object positions, starts off at 1.32 and linearly comes down to about 0.8, indicating an increase in item localisation. The classification loss, representing the accuracy of the predictions, starts off at 1.66 and falls below 1.0 in the later epochs, representing the increased capability of the model to distinguish between object classes. Distributional Focal Loss-DLF-an indicator of the bounding box placement accuracy-shows, however, a similar trend of gradual increase through training.

The validation losses are doing well. Validation box loss started at around 1.04 and went down while the model was improving its generalization on new data. The

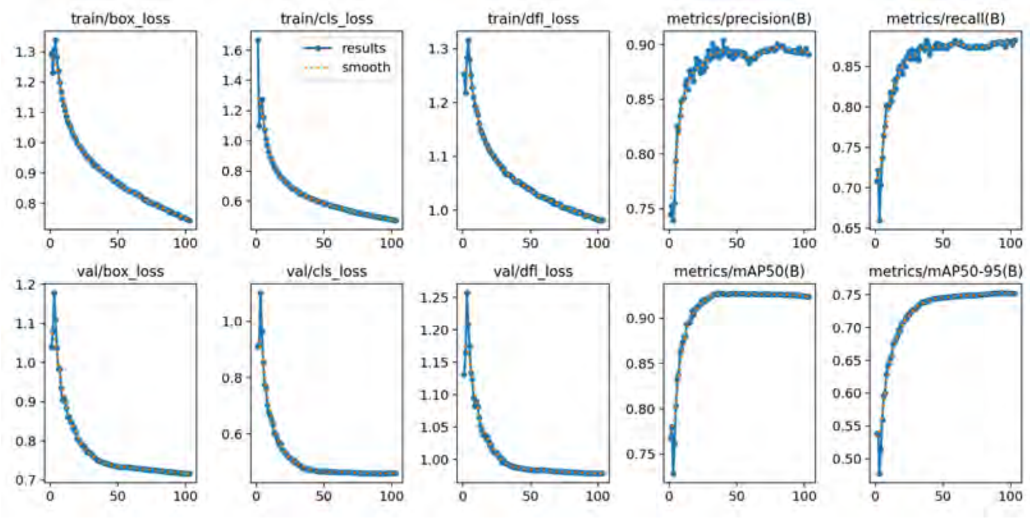


Figure 6.12: YOLOv11 Train-loss, mAP, precision and recall metrics

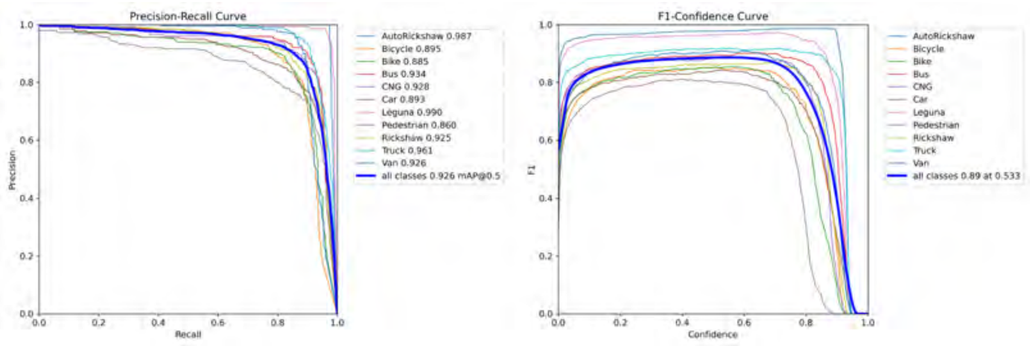


Figure 6.13: YOLOv11 Precision-Recall and f1 Curve

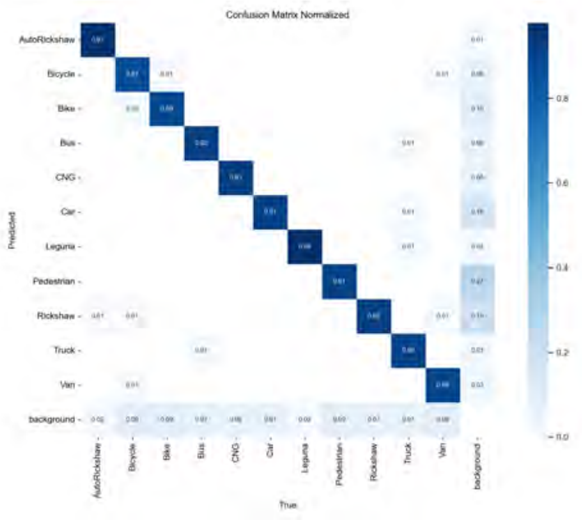


Figure 6.14: YOLOv11 confusion matrix

loss regarding validation classification improved, starting from 0.91 during the early stages and reaching about 0.85, which is an indication that the model was indeed robust for a number of object classes.

Accuracy metrics of precision had significantly improved and stabilized at about 0.896, while recall increased gradually to 0.882, indicating the model efficiency in the detection of items across the dataset. Performance metrics,-mean Average Precision at 50% IoU-means it reached 0.926, reassuringly indicating the strong detection accuracy of this model. On the other hand, the mAP50-95, reflecting a family of metrics for evaluating accuracy at different IoU thresholds, was roughly 0.752, reflecting the model's ability to generalize across different degrees of object overlap.

The confusion matrix demonstrates that class AutoRickshaw has a very high detection accuracy, with precision at 0.97 and very slight confusion with other classes. However, intersection points in other transportation categories are seen for classes like Bicycle and Pedestrian, showing avenues for improvement. The F1 score, representing a balance between precision and recall, reached 0.89 at a confidence threshold of 0.533, which shows the overall stability of the model concerning the various classes.

6.1.6 Object Detection Models Comparison

It was observed that Faster R-CNN performed less than the YOLO models both in speed and for real-time object detection. Since there is a big gap in performance, most especially when dealing with complex and fast-changing environments such as in autonomous driving, we opt to further compare the YOLO models. YOLO was more suitable for our system because of its single-pass detection ability and faster inference times; thus, the main focus of the first stage in our evaluation was to compare the different available versions of YOLO to decide on the best version for integration.

	Epochs	**mAP50**	**mAP50-95**	**Precision**	**Recall**	**F1 score**
Faster RCNN	71	0.588	0.401	0.434	0.534	0.594
YOLO v5	173	0.915	0.748	0.908	0.880	0.893
YOLO v8	147	0.930	0.764	0.903	0.898	0.898
YOLO v9	127	0.918	0.747	0.902	0.881	0.891
YOLO v11	103	0.926	0.752	0.896	0.882	0.890

Table 6.1: Comparison of object detection models based on performance metrics

In terms of mAP50 scores, YOLOv8 surpasses the others by attaining the best score of 0.930, signifying exceptional object detection precision. Conversely, YOLOv9, while very competitive, has a somewhat inferior detection performance with a mAP50 of 0.918. YOLOv11 closely resembles YOLOv8, achieving a mAP50 of 0.926, indicating robust detection capabilities but with marginally inferior overall performance. YOLOv5, though producing admirable outcomes, has the lowest mAP50 at 0.915, signifying marginally reduced accuracy in object detection relative to the other models.

It follows that YOLOv8 is also the best performer in terms of generalizing across IoU thresholds, with a score of 0.764, showing superior generalization over a more significant extent of object overlaps. Others, including YOLOv5 and YOLOv9, have comparable scores of 0.748 and 0.747, respectively. While both performed very impressively, they are less consistent than YOLOv8 in handling different IoU thresholds. YOLOv11 demonstrated a mAP50-95 of 0.752, which is marginally behind that of YOLOv8, showing a good balance but far from achieving top performance.

In terms of the F1 score, which balances Precision and Recall, YOLOv8 has an impressive 0.898. This suggests that it has the best balance in minimizing false positives while maximizing object detection. YOLOv5 achieves an F1 score of 0.893, indicating that its performance is robust but not as balanced as YOLOv8. Slightly lower F1 scores are then obtained by YOLOv9 and YOLOv11 with scores of 0.891 and 0.890, respectively. This means that the performance between them is quite similar and largely less balanced than YOLOv8.

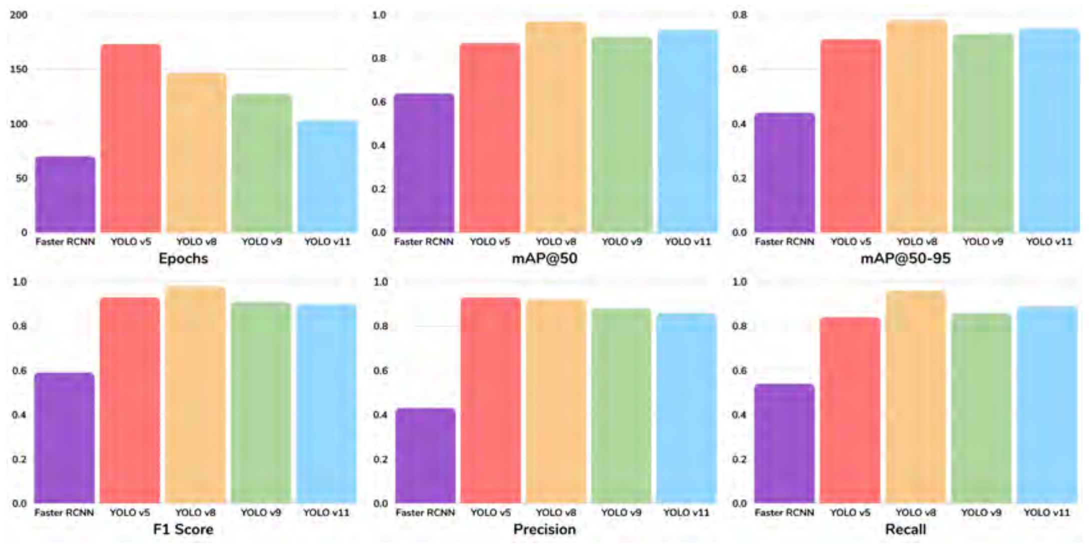


Figure 6.15: mAP@50, mAP@50-95, f1 score, precision and recall comparison of object detection models

However, all four models demonstrate impressive performance, YOLOv8 emerges as the optimal choice for integration with depth estimation models. It consistently outperforms in mAP50, mAP50-95, and Recall, indicating superior accuracy and generalization in object detection. Despite YOLOv5's slight edge in Precision, YOLOv8's overall higher performance across all metrics, particularly in balancing detection accuracy and generalization, makes YOLOv8 the optimal choice for model integration.

6.2 Depth Estimation Model Analysis

In contrast, MiDaS, MonoDepth2, and MegaDepth all have relative strengths and weaknesses. In particular, MonoDepth2 is lightweight and efficient, hence suitable for real-time applications in computationally-constrained environments. However, due to its self-supervised learning on stereo image pairs or video sequences, it re-

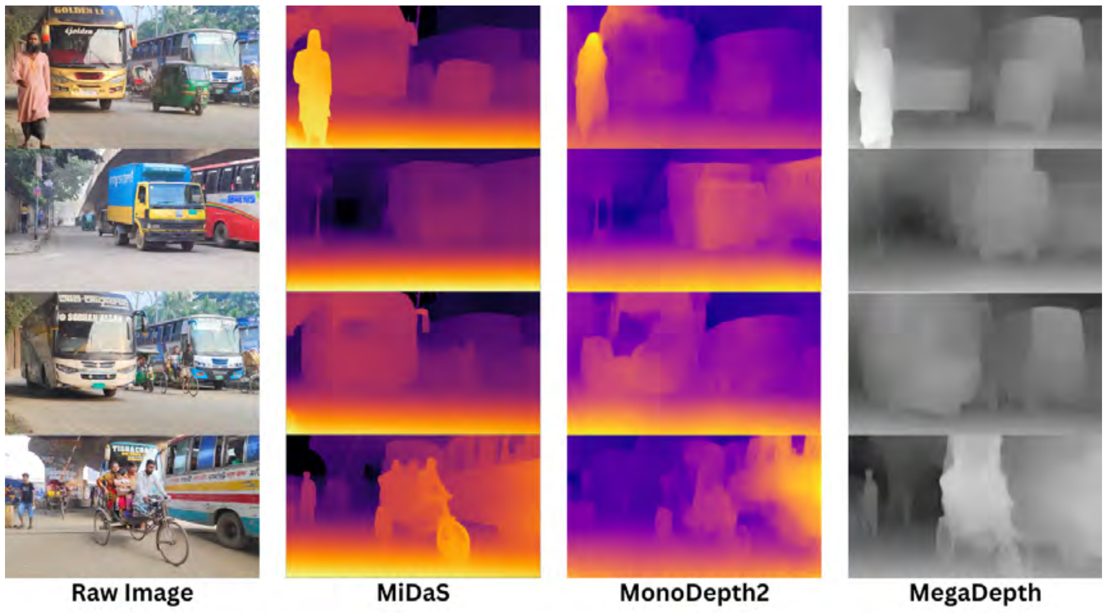


Figure 6.16: Depth estimation model comparison

stricts its generalization across diverse settings, particularly in single-image inputs. It performs comparatively worse in complicated scenarios where the expected depth map must be accurate. On the other hand, MegaDepth, although it produces high-quality depth maps from multi-views and outstanding performance outdoors in large scales, is computationally expensive and thus impracticable for real-time single-image depth estimation. Instead, it perfectly fits multi-view stereo applications but may not be efficient enough for real-time applications requiring single-image depth inference.

Contrastingly, MiDaS delivered a better performance in depth estimation and is far easier to interpret. While running MiDaS, it consistently showed better edge detection, such that the outlines of an object were better defined. Additionally, even the gradation of colors in the depth map was rather clear, from which one could immediately discern which one was closer and which was farther away. This further clarity and precision enabled MiDaS to provide more informative and useful depth maps as compared to the other two variants.

Also, MiDaS generalizes well to a large range of environments, both indoor and outdoor, since it was trained on diversified sets of datasets such as ReDWeb, MegaDepth, and ETH3D[15][10][33]. The FPN backbone allows the network to process depth at multiple scales effectively, capturing both local details and the broader scene context. As discussed above, MiDaS is chosen for the project since it gave the best overall performance, more so in view of edge detection and depth clarity. This was mainly because its ability to give clear depth information supported by good generalization across diverse scenes made it most reliable.

While MonoDepth2 was faster, and MegaDepth was outstanding for multi-view tasks, MiDaS strikes a perfect balance between speed, accuracy, and depth quality to be most suitable for applications such as autonomous driving, robotics, and augmented reality, which require accurate and reliable depth estimation.

MiDaS Model Selection: The DPT_hybrid model offers the best of both accuracy and computational speed, making it a great choice for combining with object detection models like YOLO. This hybrid model is made to work in many situations, including indoor and outdoor ones. It gives many options when estimating depth in different scenes. This wide range of uses is significant for this integration, which involves an autonomous vehicle's depth sensing in different situations.

The DPT_large model is more accurate but takes a lot of time to run on a computer, but the DPT_hybrid model is much faster. The large model usually works at a higher resolution and needs more computing power, which could slow down real-time processing, mainly when used with YOLO to find objects. On the other hand, the MiDaS_small model is made to be very light. However, its smaller size makes it less accurate at estimating depth, especially in complicated settings with many different features. It is faster, but the loss of accuracy can make it harder to find objects and measure distances accurately. So, DPT_hybrid is the best choice to integrate with an object detection model (YOLOv8) that works in real-time.

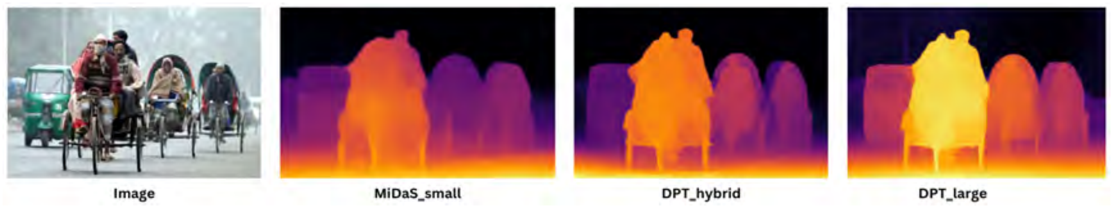


Figure 6.17: MiDaS model comparison

Chapter 7

Model Integration

7.1 Object Detection of Integrated MiDaS and YOLOv8

7.1.1 Overview of Integration

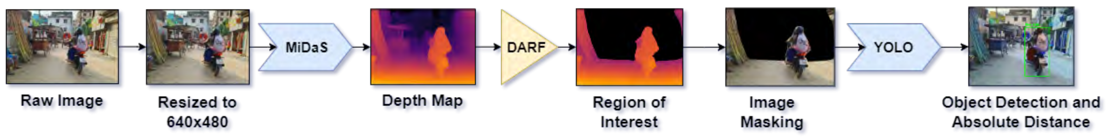


Figure 7.1: Integration overview

In the context of autonomous vehicle vision, however, this object detection across the full image is not required and would be really inefficient. Fundamentally, an autonomous vehicle needs to look at the closer regions of its environment for instantaneous decisions, such as making avoidance or navigating through dispersing traffic flow. The onus of detecting objects further away, or outside the path the vehicle will travel, unnecessarily complicates and prolongs processing, adversely impacting real-time decision making. For this reason, integration on the detection of objects only in close regions of the scene has become significant. Using depth sensing with MiDaS, we first create a depth map of the image that highlights specific portions of the image which are closer to the vehicle. Using our Depth Aware ROI Filter-DARF, we can then filter the depth map focusing on those crucial regions only. By knowing the ROIs, we can subsequently limit object detection to within those areas only, thereby bypassing the processing of the entire image.

In this way, the whole detection pipeline is optimized to have the vehicle focus on relevant objects in its immediate surroundings, further improving efficiency and safety. By intuitively reducing the scope of object detection to include only those regions necessary, this makes the workload easier. However, it also maintains in focus that the importance of timely and proper object recognition forms one of the essential real-time autonomous driving skills.

7.1.2 Integration Pipeline

In the integration process, input to the pipeline is a raw image that first passes through the depth estimation model (MiDaS) for processing to get its corresponding depth map. This depth map then provides whatever information is needed about the relative distance of objects from one another. Once the depth map is identified, further processing is performed using our state-of-the-art Depth Aware ROI Filter (DARF) which filters the region of interest (ROI). This filter refines the depth map, therefore isolating the most relevant sections of the image where objects are likely to be detected due to their proximity.

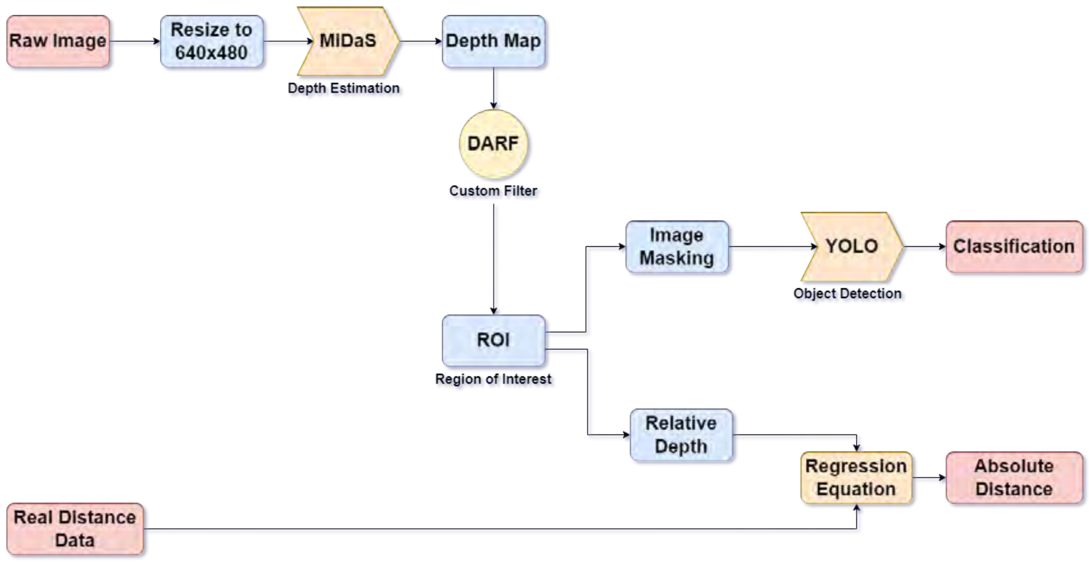


Figure 7.2: Integration pipeline

Once the region of interest has been extracted by DARF, the next step involves the cropped region of the image that is narrowed down to the critical depth areas, being fed into the YOLO model for object detection. YOLO proceeds with object detection and classification within the identified regions of interest. This depth-based selective detection will ensure that it processes only the relevant and nearby objects, hence improving efficiency and accuracy.

Concurrently, we determine the relative depth from the depth map that DARF has processed and combine it with the object detection data from YOLO to calculate the absolute distance of each detected object. This is the last step and allows us to properly calculate the real-world distances of objects in a scene and complete the process of integration. This pipeline provides frictionless flow from depth sensing to object detection, hence optimizing the detection process with depth awareness.

Image Preprocessing for MiDaS

Preprocessing was necessary when the image was given to MiDaS for depth estimation. This will ensure that the model does the job of processing the input efficiently. First, we provide the image at a resolution of 640x480. The image is then resized to 384x384, specified as the resolution for MiDaS Hybrid. This scaling ensures that the image fits within the model's required input dimensions and accounts for any difference in aspect ratio by adding padding after scaling if needed to prevent distortion. The pixel values are then further normalized between 0 to 1 inclusive. Since the normal images have pixel values between 0 and 255, normalizing the pixel values by dividing 255 is required. If it is not in that format, the image is then converted to the RGB color space after normalization, since MiDaS only operates on RGB images.

Depth Map Estimation

MiDaS makes use of a powerful convolutional neural network which has the ability to extract all the necessary features of an image. Such features are crucial, including some critical information such as edges, textures, and object shapes, making it easier for MiDaS to build a hierarchical representation of the scene. Specific to MiDaS, it is particularly useful because of its ability to glean global context from one image. While traditional depth estimation requires stereo imagery or multiple views, MiDaS makes estimates on depth relations between objects by leveraging patterns learned across diverse landscapes. It allows for an approximation of which regions in the image are closer or further; it simply relies on the visual cues of object size, position, and shading. Once the features have been extracted, MiDaS predicts the comparative depth of each pixel within the image. This gives as output a depth map where warmer colors-reds and yellows-indicate things closer, and cooler colors-blues-indicate objects that are further away. Such a depth map thus provides a per-pixel representation of the structure of the scene depicted, allowing for preliminary identification of regions of interest on which object detection and further analysis shall be performed.

Depth Aware ROI Filter (DARF)

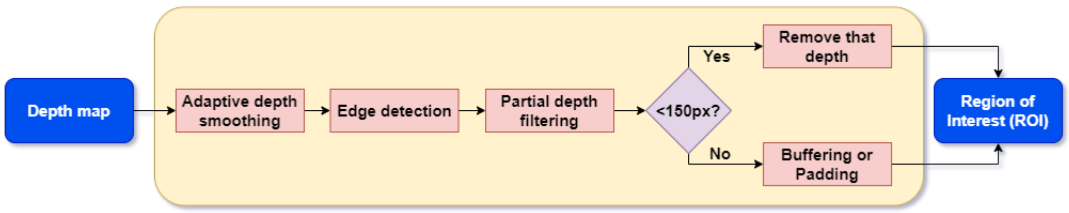


Figure 7.3: DARF flowchart

In building the Depth Aware ROI Filter (DARF), the process involves a series of steps aimed at getting the region of interest (ROI) for object detection based on depth information. The goal is to focus detection on relevant areas in an image, enhancing both accuracy and efficiency. A detailed description of each step involved in constructing the DARF follows:

1. Adaptive Depth Smoothing: The first step involves taking the depth map from MiDaS and applying adaptive depth smoothing on it. This technique smoothens the depth map dynamically concerning local variations of depth. Smoothing helps reduce the noise and small fluctuations in the depth values. It is a crucial step to make the detection process more reliable. This adaptive smoothing takes into account the variance in the range, so this gives a good representation of the objects in both the near and far ranges. It maintains depth discontinuities between objects, such as sharp edges, while continuing to suppress noise across smooth transitional regions.

2. Edge Detection: Smoothing is followed by the usage of edge detection on the depth map. This step is helpful in obtaining a much finer identification of object boundaries when the obtained depth values vary significantly between adjacent pixels. Using edge detection, the outlines of object contours within the considered scene can be drawn, representing the fact that the correct segmentation of the ROI regarding the real objects takes place. It works to further identify each distinct object within the depth map, but more importantly, it carves out the front objects from the background and enhances focus on the relevant areas.

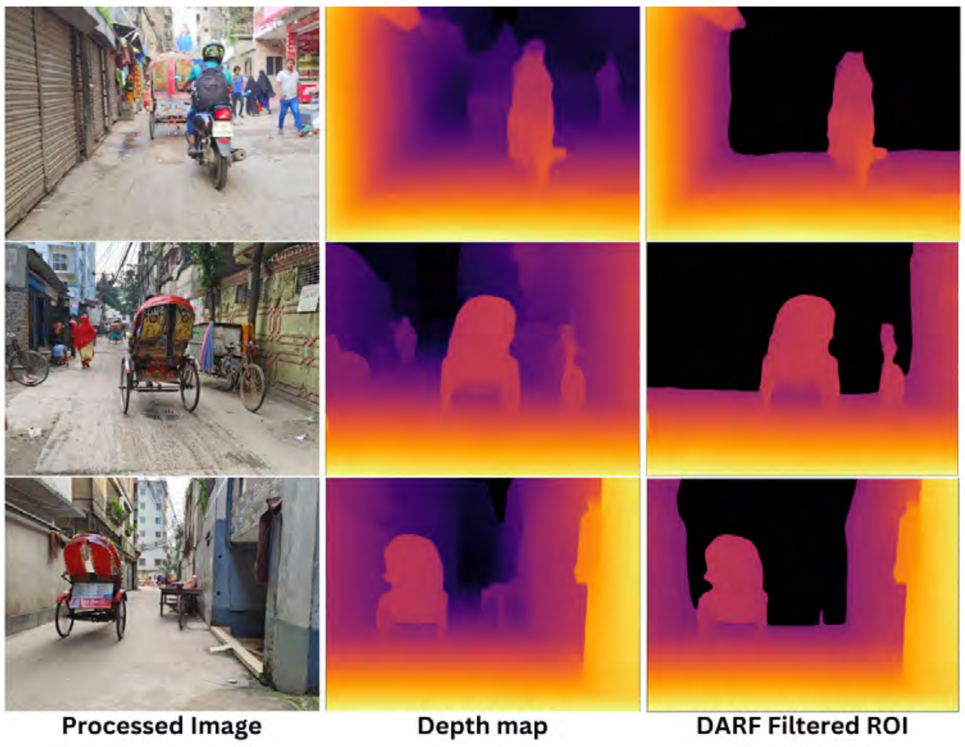


Figure 7.4: DARF filtered region of interest (ROI)

3. Partial Depth Filtering: Once the edges are computed, the filter analyzes partial depths in the depth map. It is mainly concentrated on small depth or incomplete depth. The idea here is to remove fragments of objects that have less than 150 pixels in size. These small fragments usually originate from errors in the estimation of depth or minor inconsistencies in the depth map and do not add value to the detection task. Filtering out objects depth smaller than 150 pixels, the system doesn't make use of any more resources than necessary on irrelevant data but focuses only on the meaningful parts of the image.

4. Buffering/Padding: The partial object greater than 150 pixels, which is not fully enclosed within the ROI, will have buffers or padding applied to it. This helps ensure that larger objects, perhaps cropped out by depth thresholds, have enough coverage within the ROI. Extra space is added along the edges of the detected object's boundaries to account for missing parts and provide room to consider the whole object during the detection phase. It enhances object completeness inside the ROI; hence, making the final detection more accurate.

YOLO on ROI Masked Image

By using DARF once the ROI is defined, the original image is cropped or masked in order to focus only on the relevant areas to the depth. These will be the locations where object detection will be made possible by YOLO. In doing so, the system restricts the operation of YOLO to the particular regions where significant objects are most likely to exist. Depth map plays a very important role in selecting this region, giving more importance to those places where objects are nearer and clearer concerning depth perception. In such cases, objects that have a clear boundary in the depth map and are nearer to the camera are given more priority. The object detection by YOLO will take place within this filtered region only. This selective processing keeps YOLO focused on the more relevant objects of interest, which saves time in object detection and makes it more suitable for the demands of real-time applications.

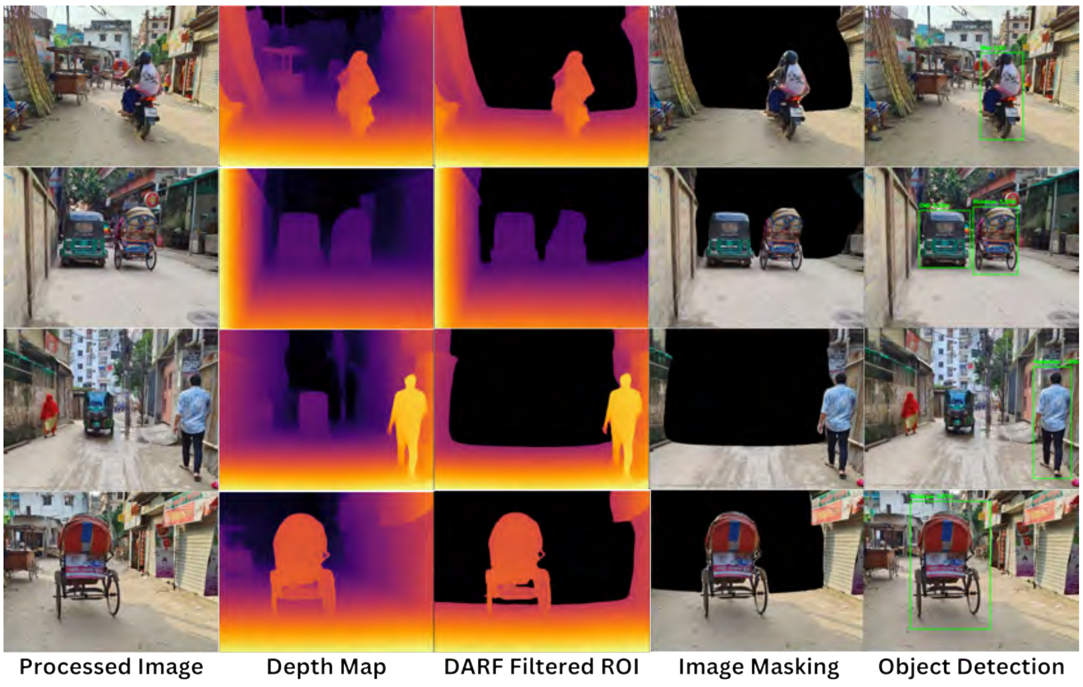


Figure 7.5: Object detection and absolute distance through integrated model

7.2 Absolute Distance from ROI

7.2.1 Relative Depth Calculation from ROI

Extraction of Depth Values from ROI: Selecting the area of the depth map that aligns with the DARF filtered region where the object is detected. Then the object depth value is defined by a bounding box formed by the top-left and bottom-right corners surrounding the detected object. Extract the depth values from the depth map for this area.

$$\text{depth\_value} = \text{depth\_map}[x_{\text{min}} : x_{\text{max}}, y_{\text{min}} : y_{\text{max}}]$$

Mean Depth: This facilitates an average depth for the object.

$$\text{mean\_depth} = \frac{1}{N}\sum_{i=1}^{N} \text{depth\_value}_i$$

Here, N = Number of pixels

Median Depth: Median depth is necessary to remove noise and small elements of background.

$$\text{median\_depth} = \text{Median}(\text{depth\_value})$$

Max Depth: The closest point of the depth map which is the largest depth value of the image.

Min Depth: The farest point of the depth map which is the smallest depth value of the image.

Relative Depth: From mean depth, median depth, max depth and mean depth the relative depth can be calculated.

$$\begin{aligned}\text{Relative Depth} = {} & w_1 \cdot \text{mean\_depth} + w_2 \cdot \text{median\_depth} \\ & + w_3 \cdot \text{min\_depth} + w_4 \cdot \text{max\_depth}\end{aligned}$$

Here, $w_1 = 0.5$, $w_2 = 0.2$, $w_3 = 0.15$, and $w_4 = 0.15$.

Normalization of Relative Depth: This ensures the consistency of depth values in various scenes.

$$\text{Normalized Depth} = \frac{\text{depth\_value} - \min(\text{depth\_map})}{\max(\text{depth\_map}) - \min(\text{depth\_map})}$$

In short, to determine the relative depth of an object, we obtain the depth values from the DARF filtered region within the depth map produced by MiDaS. Subsequently, we calculate a weighted amalgamation of the mean, median, minimum, and maximum depth data, assigning greater weight to the mean depth for a better estimate. This method offers a comprehensive relative depth assessment that considers the object's mean depth as well as its nearest and farthest areas. The outcome is a reasonable estimation of the object's location.

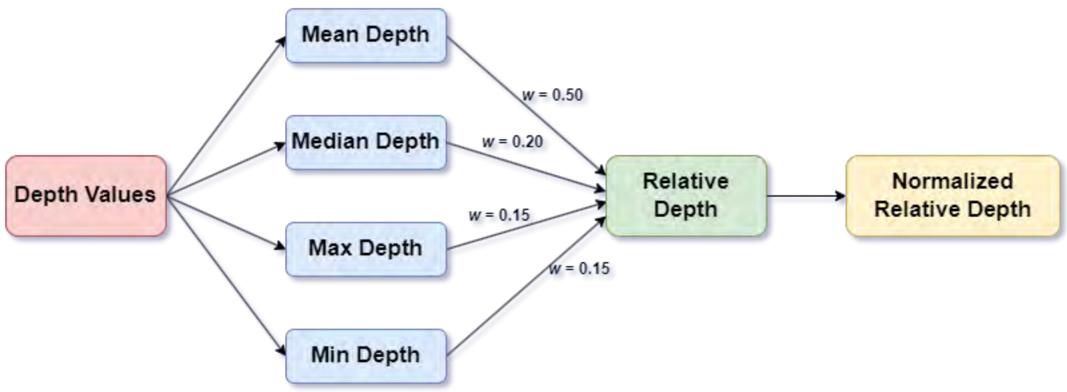


Figure 7.6: Relative depth calculation process

7.2.2 Relationship Between Relative Depth and Real Distance

Understanding the relationship between relative depth and real-world distance is must in pursuing the improvement of accuracy in our depth-aware object detection system. An experiment was conducted where data from 60 objects for which their respective real distances were measured at 2 meters, 3 meters, 4 meters, and 5 meters was collected. For each object, we computed the corresponding relative depth using the depth map of ROI generated by DARF. We plot the data measured on a graph, with relative depth on the x-axis and real distance on the y-axis. In this trend, we will be able to see the variation between the two variables of relative depth and real distance. By plotting the data measured, we attained a general trend with a scattered distribution and used regression analysis to attain an equation describing the best fit for the relationship.

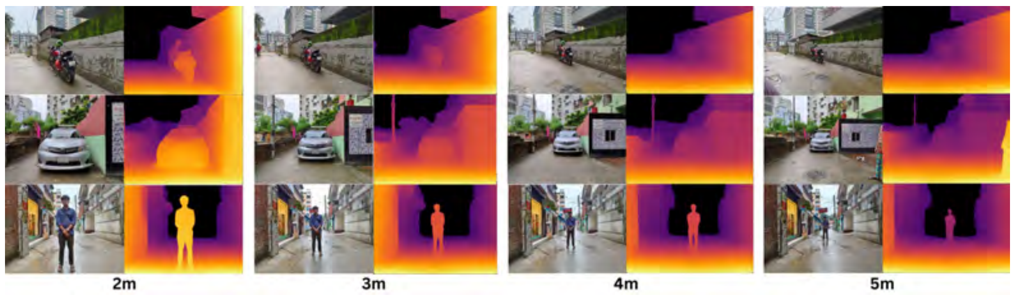


Figure 7.7: Relative depth measurement for respective real distance

7.2.3 Determining the Equation

To further investigate this, a scatter plot of the relative depth on the x-axis and real distance on the y-axis was plotted to which various regression models were fitted in order to explore the best fit. Each kind of regression gives another different way of understanding and helps to determine the most accurate model for estimation of real distance from a given relative depth. Using these different regression models we can

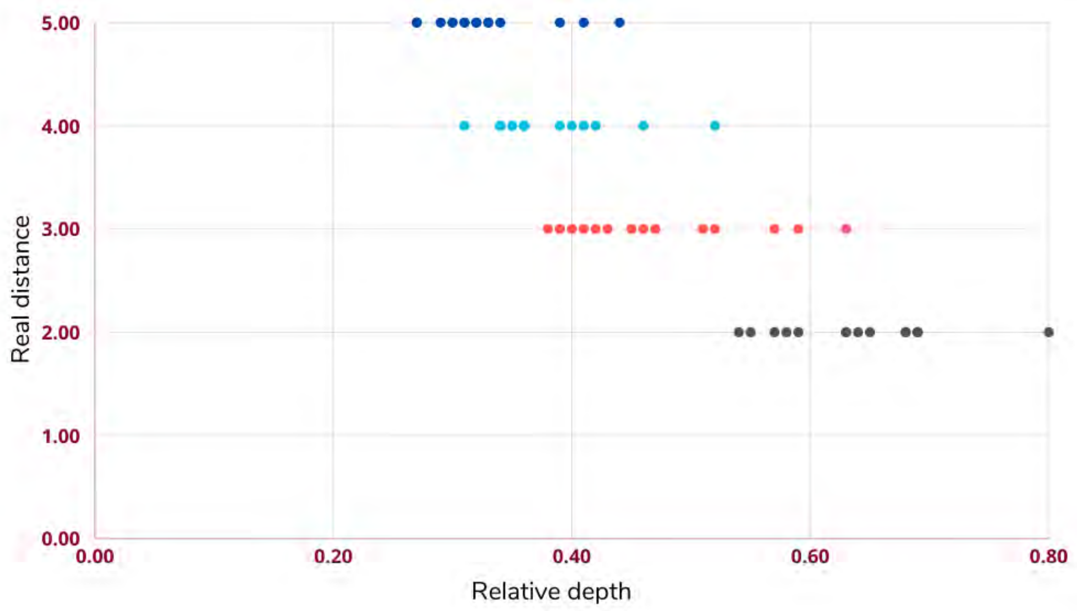


Figure 7.8: Relative depth vs real distance graph

analyze the relationship of relative depth and real distance from various viewpoints, with each regression type teaching something about the underlying patterns in data. Which is the best model needs to be judged by which can fit the observed data most closely, whereas among them, we assess their performance based on error rates, residuals, and general goodness-of-fit measures. It will be, eventually, the one that will allow us to convert the relative depths to real-world distances with great precision, an indispensable step in refining the precision and further improvement of the overall performance sought for our depth-aware objection detection system.

Linear Regression

Linear regression assumes that there exists a straightforward, proportional relationship between relative depth and real distance. So we fit a straight-line equation to our data points of the form:

$$y = mx + c$$

where y is the real distance, x the relative depth, m the slope of the line, and c the intercept. That is an easy and efficient approach in the case of an approximately linear relationship between the variables. However, in cases of more complex or nonlinear relationships such as that between relative depth and real distance linear regression may fail to provide the best fit.

In our case, we get the equation of linear regression-

$$\text{Absolute Distance} = -7.18 \cdot \text{Relative Depth} + 6.77$$

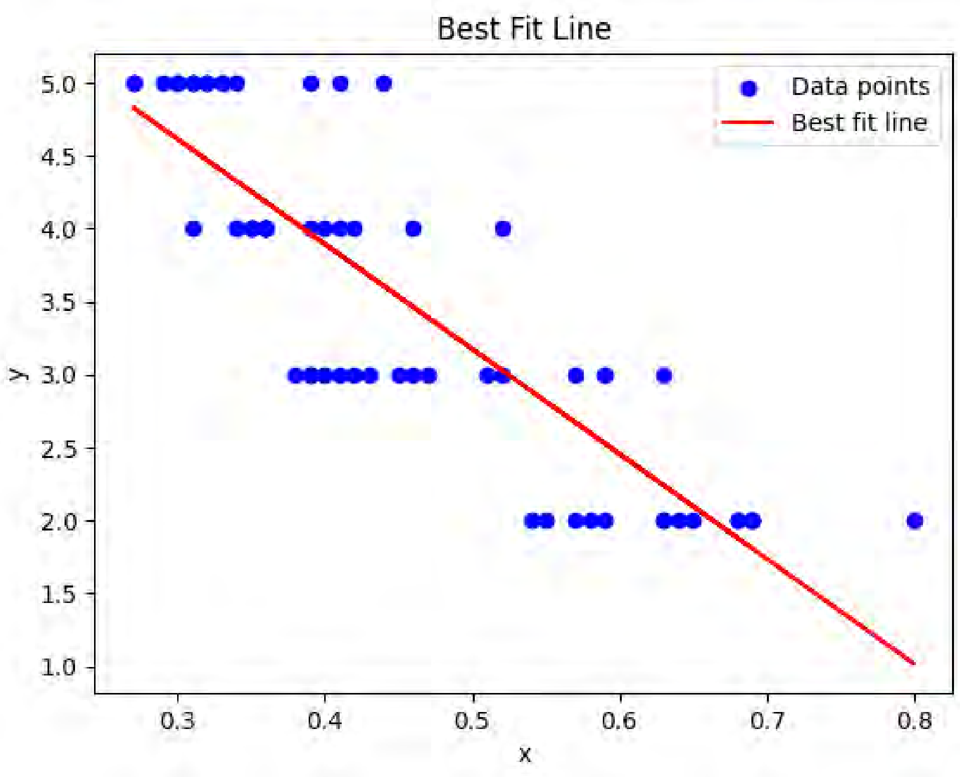


Figure 7.9: Linear regression for our data

Polynomial Regression

In order to capture more complex nonlinear relationships between relative depth and real distance, we applied polynomial regression. This methodology fits a curve to the data by adding higher-order terms to the equation such as:

$$y = ax^2 + bx + c$$

This has the capability of capturing curvatures in the data and thus is more flexible than linear regression. Polynomial regression will be very useful when the pattern in the data has a significant curvature; this is because it could fix such a pattern with higher precision. However, care should be taken in choosing the degree of the polynomial because high-degree polynomials become extremely sensitive even to minor wobbling in the data.

In our case, we get the equation of polynomial regression-

$$\text{Absolute Distance} = 11.68 \cdot \text{Relative Depth}^2 - 18.80 \cdot \text{Relative Depth} + 9.43$$

Exponential Regression

Exponential regression is used when the relation between relative depth and real distance changes at a constant multiplicative rate. It takes the form of the model:

$$y = ae^{bx}$$

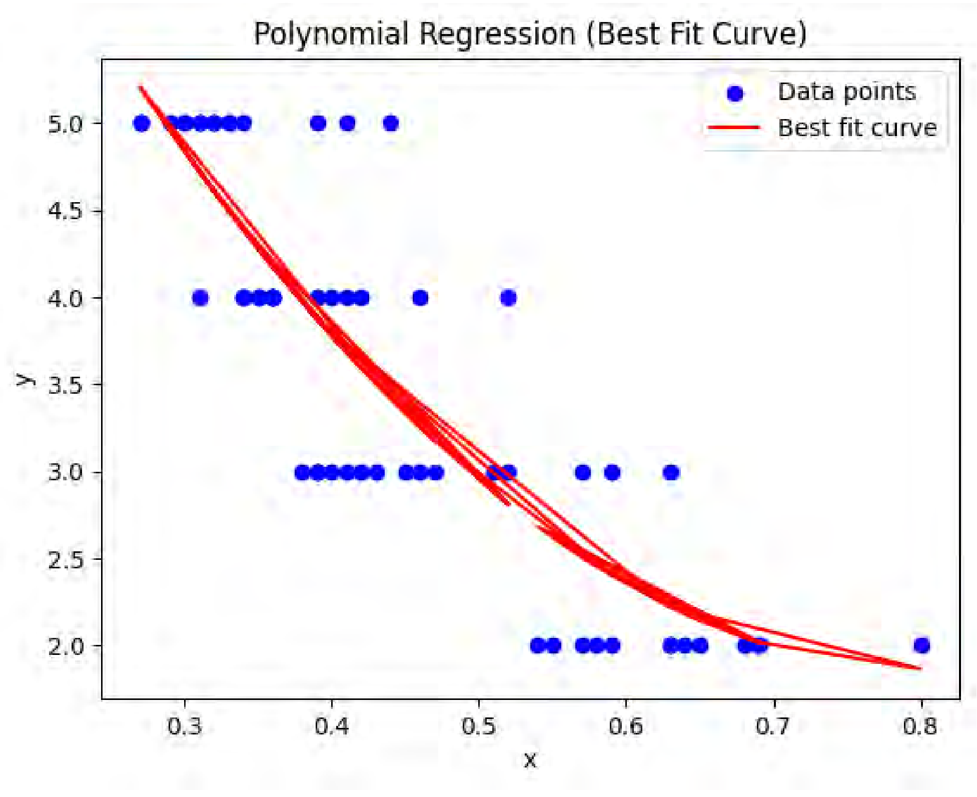


Figure 7.10: Polynomial regression for our data

where y is the real distance, x is the relative depth, a is a scaling constant, and b controls the rate of exponential growth or decay. This is an effective way when the rate-of-change accelerates or decelerates rapidly; the distances grow exponentially as the relative depth changes. No exponential growth/decay is observed systematically and hence the application of exponential regression could not fit our data best.

In our case, we get the equation of exponential regression-

$$\text{Absolute Distance} = 9.68 \cdot e^{-2.34 \cdot \text{Relative Depth}}$$

Logarithmic Regression

Logarithmic regression models a relationship in which the rate of change in real distance slows down as relative depth increases. The model takes the form of:

$$y = a \ln(x) + b$$

Such regression is useful when data suggests that real distance increases rapidly for low values of relative depth while increasing ever more slowly as the values of the relative depths increase. Logarithmic regression is able to model such decelerating growth, which often arises in natural processes. It works particularly well when the data follows diminishing returns, where, for each increment in relative depth, the real distance increases by a smaller amount.

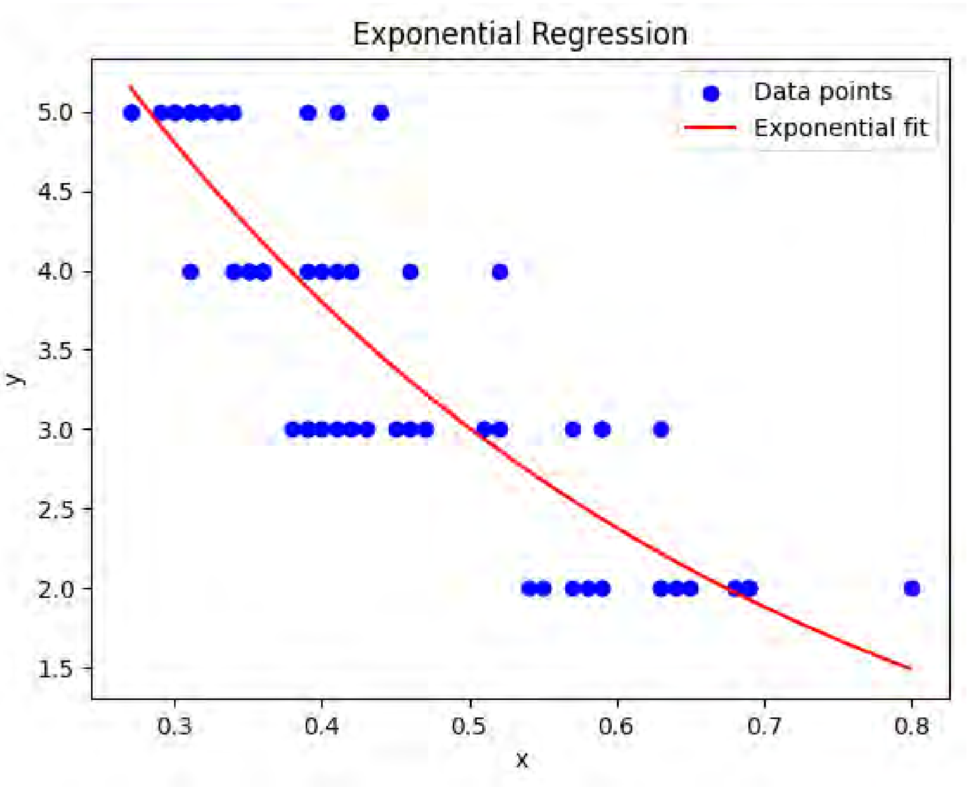


Figure 7.11: Exponential regression for our data

In our case, we get the equation of logarithmic regression-

$$\text{Absolute Distance} = 0.66 - 3.43 \cdot \ln(\text{Relative Depth})$$

Power Regression

Power regression applies to modeling proportional relationships where real distance increases with a constant factor as the relative depth increases. It assumes the form:

$$y = ax^b$$

In this model, a and b are constants, with the exponent controlling curvature. Power regression is very useful to model when there is a straight scaling relationship between variables. For instance, if doubling the relative depth goes hand in hand with the proportionate increase in real distance. It finds usual application in physical phenomena where scaling laws are in effect; therefore, this model should be a good candidate for estimating depth based on a real-world environment.

In our case, we get the equation of logarithmic regression-

$$\text{Absolute Distance} = 1.46 \cdot \text{Relative Depth}^{-1.01}$$

Video demonstration of integrated model: https://youtu.be/YlU3-R0OlJE

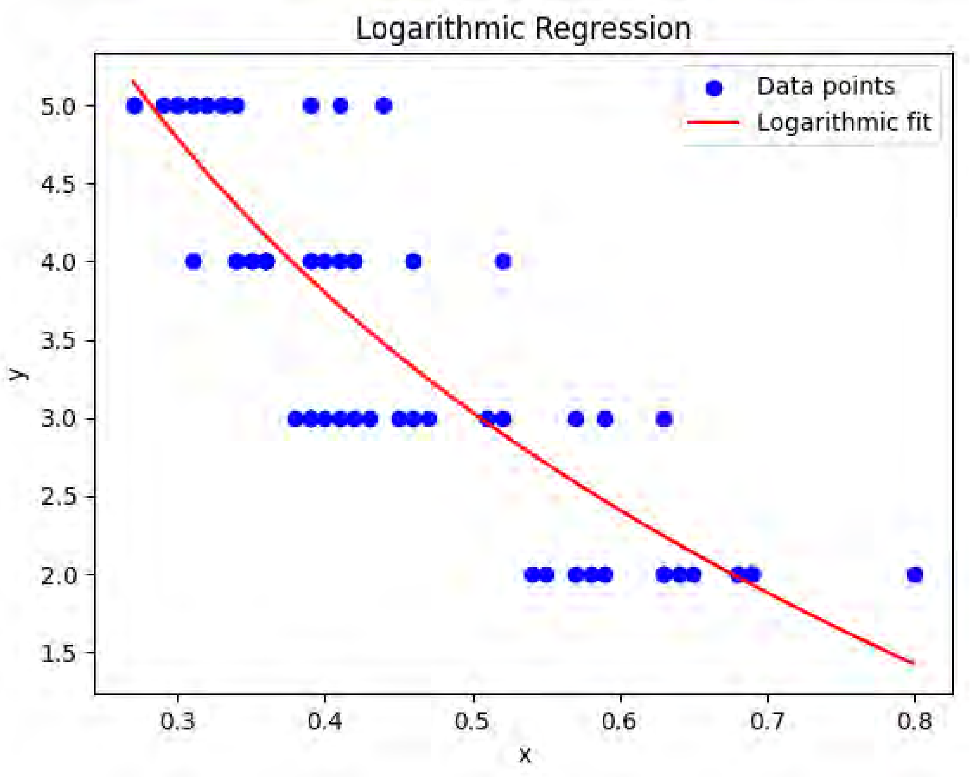


Figure 7.12: Logarithmic regression for our data

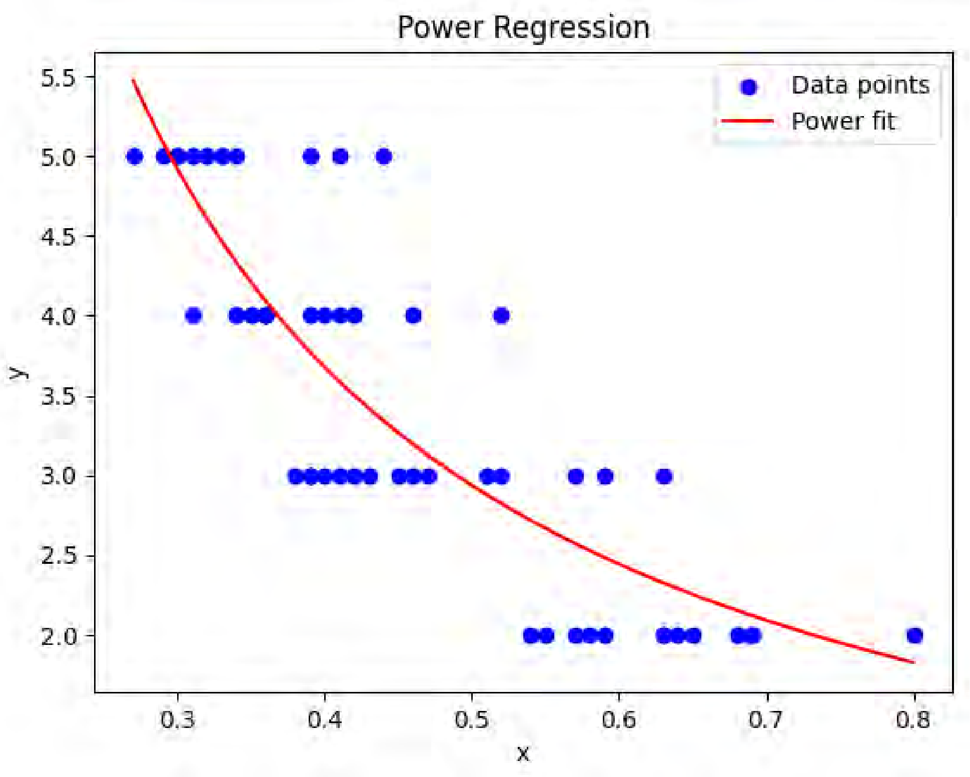


Figure 7.13: Power regression for our data

Chapter 8

Integrated Model Evaluation and Analysis

8.1 DARF Evaluation

To evaluate the performance evaluation of our integrated model which restricts object detection in the Depth Aware ROI Filtered region, we tested one hundred images of different road view scenarios. Then, YOLO was applied on full images and the time required for processing the object detection was measured. Also, YOLO was applied only in the DARF filtered region of the images and the time required for processing the object detection was measured. After analyzing the processing time, the object detection in the DARF filtered region is 85.22% faster than object detection in a full image. This huge improvement of almost seven times faster shows that DARF considers bringing a huge speed in object detection for an autonomous vehicle, especially in real time.

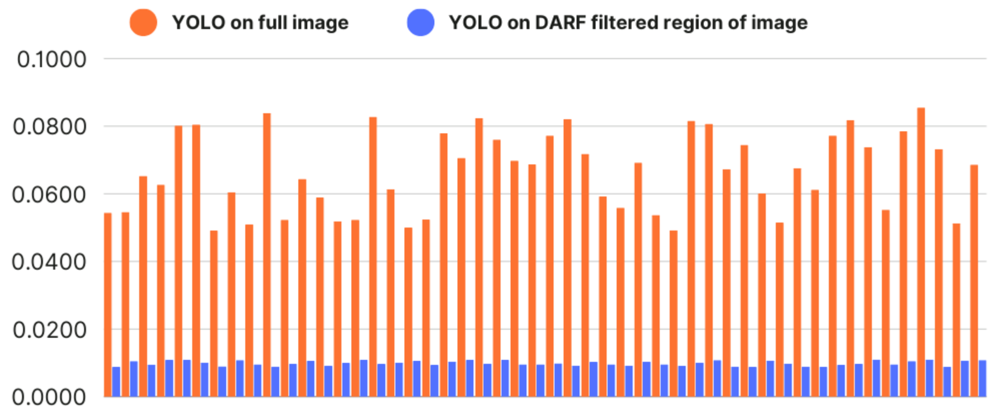


Figure 8.1: Processing time graph of YOLO on full image and YOLO on DARF filterd region (ROI)

However, considering the whole system performance in total, it is about one and a half times slower than traditional YOLO object detection alone. In spite of all that, our system is doing not only object detection but also providing depth data, which is crucial for autonomous vehicles. This depth information enables many features apart from object detection, including path planning, lane and road surface understanding, parking assistance, and 3D Mapping with SLAM. However, depth information is required for safe and efficient navigation of autonomous vehicles.

Further, the incorporation of DARF elevates image processing like occlusion elimination, gesture detection of pedestrians, traffic sign and signal recognition, and object tracking by addressing only those areas of the image where the action has to be captured. Further, DARF allows parallel image processings to be faster and more efficient as every processing time will be greatly reduced. That is to say, DARF greatly reduces the total processing time in image processing and hence is an effective approach toward improving the efficiency in autonomous vehicle systems.

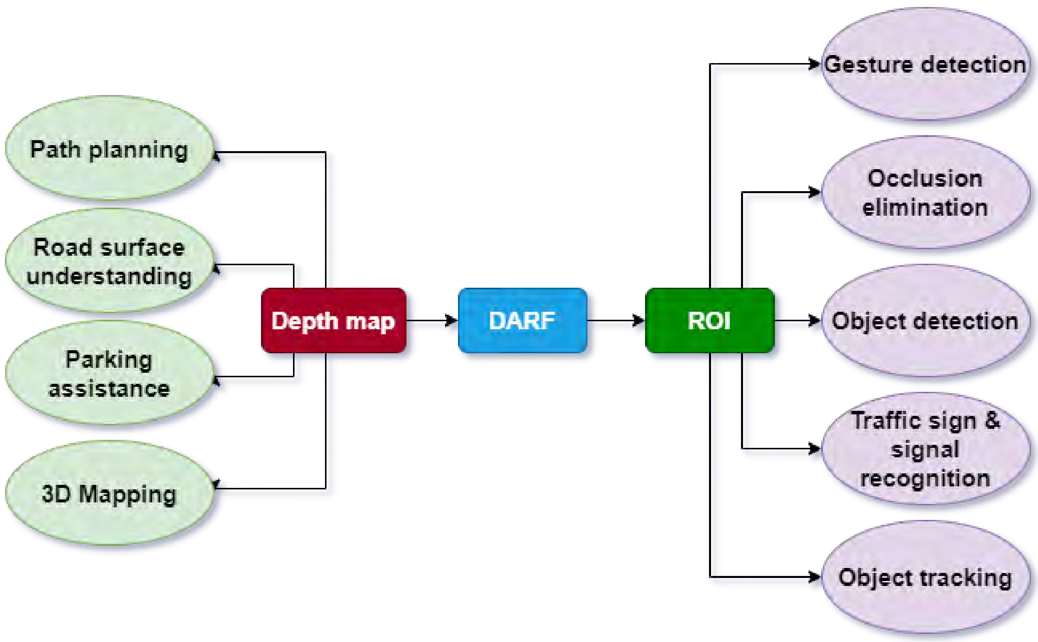


Figure 8.2: Scopes of DARF and depth map

8.2 Absolute Distance Evaluation

In this section we are going to evaluate the various regression models in estimating real distance based on relative depth. Here, we have applied five different regression methods: Linear, Polynomial, Exponential, Logarithmic, and Power regression. Each of these was tested for various real distances (2m, 3m, 4m, and 5m) in order to analyze how each model expresses the relationship between relative depth and real-world measurements. The error of each of the models at each distance was calculated, and then the average error of each regression type was computed to compare their overall performance in the task. The following table gives the error values for each of the regression models applied, and it may indicate which one of these performs a better job of estimating actual distance from relative depth estimates.

	2m	**3m**	**4m**	**5m**	**Average**
Linear	20.18	20.51	7.50	11.80	14.99
Polynomial	12.43	18.27	9.61	10.31	12.65
Exponential	14.62	18.48	9.09	10.52	13.18
Logarithmic	15.66	18.37	8.84	10.61	13.37
Power	16.80	15.87	9.47	11.25	13.35

Table 8.1: Error (percentage) comparison of regression models for different distances

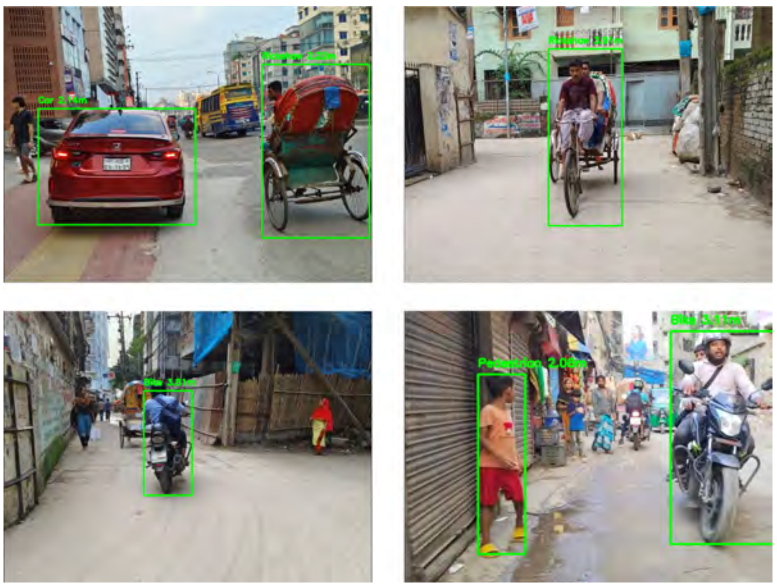


Figure 8.3: Absolute distance from DARF filtered region

The biggest errors are given by Linear Regression, which at 2m and 3m are 20.18 and 20.51, respectively. This error decreases with the increase in distance, showing 7.50 at 4m and 11.80 at 5m. However, an average error of 14.99 indicates that a simple linear approach does not model well the relationship, especially for closer objects. However, Polynomial Regression has the best results with much smaller errors at the shorter distances of 2m and 3m, at 12.43 and 18.27 respectively, while keeping these low at 4m and 5m. The overall average for this model is 12.65, the lowest among all models tested. This means the higher-order fit of the polynomial model better captures the depth-to-distance relationship, accommodating non-linearity in the data.

Exponential Regression has moderate errors across the board: 14.62 for 2m, 18.48 for 3m, while for 4m and 5m these are lower, at 9.09 and 10.52. The average error is 13.18, slightly higher than polynomial regression but better than linear regression, suggesting that while exponential growth captures some of the data, it doesn't fully capture the variation at shorter distances. Furthermore, Logarithmic Regression follows a similar trend, with errors ranging from 8.84 at 4m to 18.37 at 3m. Its average error is 13.37, which is decent but not as good as polynomial regression. The logarithmic model may capture some trends related to depth but doesn't fully model the distance-depth relationship, particularly at closer ranges.

Finally, Power Regression performs relatively well, with errors of 9.47 and 11.25 at

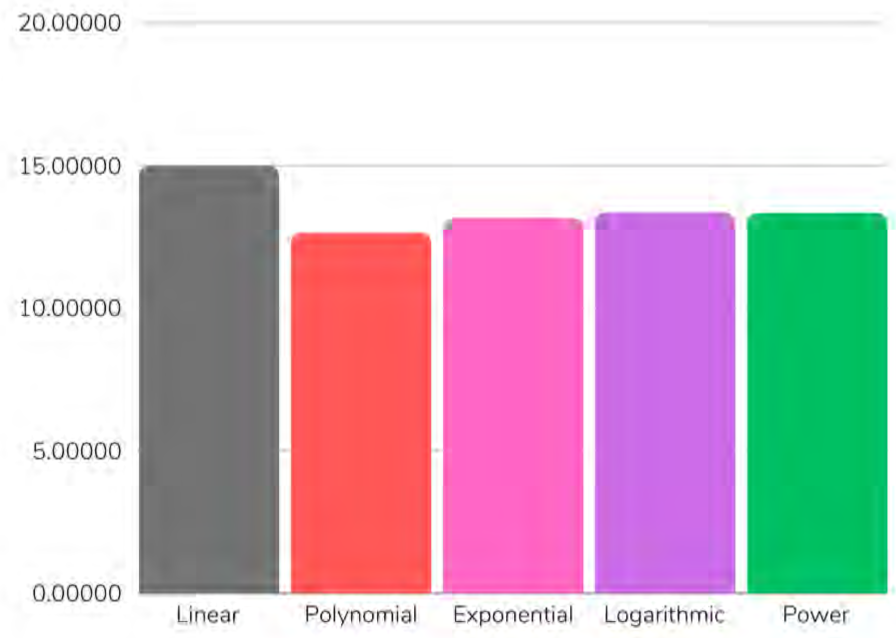


Figure 8.4: Error comparison graph of different regression models

4m and 5m respectively. While it has higher errors than polynomial regression at 2m and 3m, the overall average error of 13.35 suggests that power regression models the depth-to-distance relationship reasonably but lacks the precision of the polynomial model.

In summary, Polynomial Regression has the lowest overall error and provides the best fit for estimating real distance based on relative depth. It effectively captures the nonlinear relationship between depth and distance. The linear model performs the worst, with significant errors at shorter distances, and is not recommended. Exponential, logarithmic, and power regressions offer moderate performance, but none match the accuracy of the polynomial model, which should be chosen for further work on this depth estimation system.

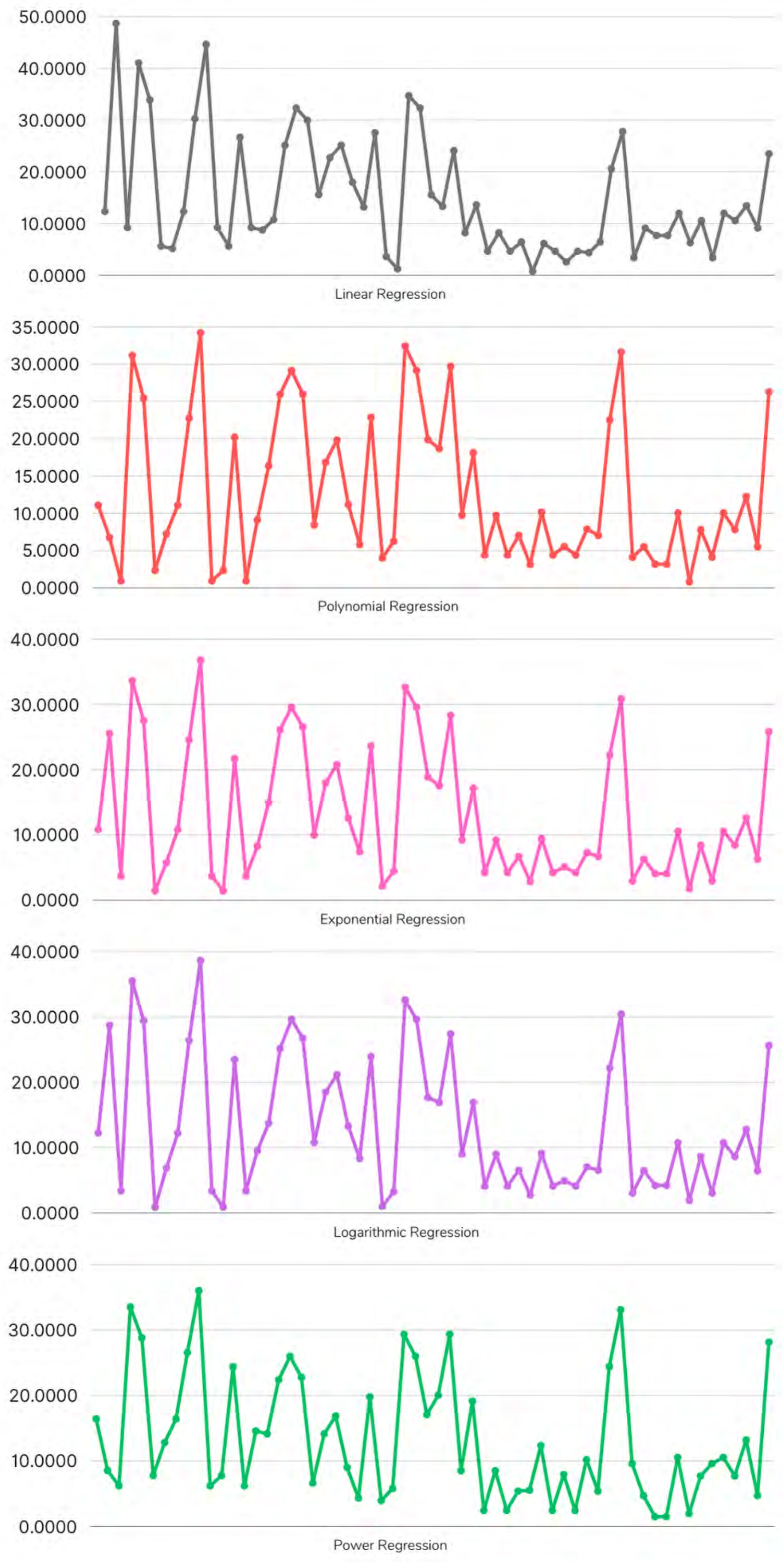


Figure 8.5: Error graph of different regression models

Chapter 9

Hardware Specifications

Due to the size of our dataset along with the usage of heavy models, hardware implementation was necessary. We used Intel Core i9 13900K which is a 24-Core Processor, Nvidia RTX 4090 graphics card along with 64GB RAM. This graphics card has 24GB VRAM. According to our graphics processing unit (GPU), we installed the compatible CUDA version 12.6 and cuDNN Version 9.5.0. RTX 4090 comes with a substantial amount of cuda cores which is 16,384. It also significantly reduces training times and improves overall performance compared to less powerful GPUs and CPUs. We used a deep learning framework such as Tensorflow which is optimized for GPU acceleration. Moreover, we captured the raw images using both Samsung GN5 and Sony IMX772 cameras with focal lengths of 24mm and 26mm, respectively. The captured images are then processed to adapt to our model requirements so that they can match the depth estimation and object detection pipeline.

Chapter 10

Future Work

In the near future, we would like to extend tracking of moving objects within this integration such that the autonomous vehicle can make real-time decisions based on moving objects. This will be more relevant in road scenarios, where most of the objects are in general motion. A correct estimation of the location and speed of a moving object will help the vehicle to find a more appropriate way to proceed safely and efficiently. Along with this, we would work on reducing the error of absolute distance estimation by collecting more data so that autonomous vehicles are able to find the distance of objects more precisely. We will also increase the dataset by having data of more and more cities of Bangladesh. Further, we intend to build in a GUI that will make our task much easier in visualizing the results of model integration. Going ahead, our study is targeted at enhancing the system to cope with the more complex road scenarios of Bangladesh, where a variety of unique conditions and challenges require robust adaptable solutions.

Chapter 11

Conclusion

The paper concludes by introducing a new approach in combining depth sensing and object detection, particularly addressing some of the challenges unique to the demands of autonomous vehicles navigating through potentially complex road environments in Bangladesh. In our study, we applied MiDaS for depth estimation and YOLO for object detection while proposing DARF to optimize object detection by focusing only on the relevant depth regions, thereby improving efficiency in its processing. Our studies showed that this method accelerates object detection processing by 85.22%, compared with the common full-image detection methods, while it also yields useful depth information for tasks that involve path planning, obstacle avoidance, and real-time decision making in self-driving cars. We further carefully explored several regression models to calculate the relative depth-to-real distance scale factor. With their diverse ranges of vehicles and road conditions, including Bangladesh road data further strengthens the applicability of our approach to real-world scenarios in this region. We focus on concentration with regard to road views from different cities and varied weather conditions, hence dealing with a comprehensive dataset that well reflects the intricacies of the local driving environment.

Bibliography

[1] B. Tippetts, D. J. Lee, K. Lillywhite, and J. Archibald, "Review of stereo vision algorithms and their suitability for resource-limited systems," *Journal of Real-Time Image Processing*, vol. 11, no. 1, pp. 5–25, Jan. 2013. DOI: 10.1007/s11554-012-0313-2. [Online]. Available: https://doi.org/10.1007/s11554-012-0313-2.

[2] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, Dec. 2015. [Online]. Available: https://arxiv.org/abs/1512.03385v1.

[3] C. Godard, M. A. Oisin, and G. J. Brostow, *Unsupervised Monocular Depth Estimation with Left-Right Consistency*, Sep. 2016. [Online]. Available: https://arxiv.org/abs/1609.03677v3.

[4] M. Mancini, G. Costante, P. Valigi, and T. A. Ciarfuglia, "Fast robust monocular depth estimation for obstacle detection with fully convolutional networks," *IEEE*, Oct. 2016. DOI: 10.1109/iros.2016.7759632. [Online]. Available: https://doi.org/10.1109/iros.2016.7759632.

[5] A. Newell, K. Yang, and J. Deng, *Stacked hourglass networks for human pose estimation*, Mar. 2016. [Online]. Available: https://arxiv.org/abs/1603.06937v2.

[6] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, *Pixelwise View selection for unstructured Multi-View Stereo*. Jan. 2016, pp. 501–518. DOI: 10.1007/978-3-319-46487-9\{_}31. [Online]. Available: https://doi.org/10.1007/978-3-319-46487-9_31.

[7] W. Hartmann, S. Galliani, M. Havlena, L. Van Gool, and K. Schindler, "Learned multi-patch similarity," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[8] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-Time dense monocular SLAM with learned depth prediction," *IEEE*, Jul. 2017. DOI: 10.1109/cvpr.2017.695. [Online]. Available: https://doi.org/10.1109/cvpr.2017.695.

[9] C. Godard, M. A. Oisin, M. Firman, and G. Brostow, *Digging into Self-Supervised Monocular depth estimation*, Jun. 2018. [Online]. Available: https://arxiv.org/abs/1806.01260v4.

[10] Z. Li and N. Snavely, "Megadepth: Learning single-view depth prediction from internet photos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[11] Z. Li and N. Snavely, *MegaDepth: Learning Single-View Depth Prediction from Internet Photos*, Apr. 2018. [Online]. Available: https://arxiv.org/abs/1804.00607v4.

[12] Y. Ren, C. Zhu, and S. Xiao, “Object detection based on Fast/Faster RCNN employing fully convolutional architectures,” *Mathematical Problems in Engineering*, vol. 2018, pp. 1–7, Jan. 2018. DOI: 10.1155/2018/3598316. [Online]. Available: https://doi.org/10.1155/2018/3598316.

[13] J. Chang and G. Wetzstein, “Deep Optics for Monocular Depth Estimation and 3D Object Detection,” *IEEE*, Oct. 2019. DOI: 10.1109/iccv.2019.01029. [Online]. Available: https://doi.org/10.1109/iccv.2019.01029.

[14] Z. Liu, M. Arief, and D. Zhao, “Where should we place LiDARs on the autonomous vehicle? - An optimal design approach,” *IEEE*, May 2019. DOI: 10.1109/icra.2019.8793619. [Online]. Available: https://doi.org/10.1109/icra.2019.8793619.

[15] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, *Towards robust monocular depth estimation: mixing datasets for zero-shot cross-dataset transfer*, Jul. 2019. [Online]. Available: https://arxiv.org/abs/1907.01341v3.

[16] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, “FastDepth: Fast Monocular Depth Estimation on Embedded Systems,” *IEEE*, May 2019. DOI: 10.1109/icra.2019.8794182. [Online]. Available: https://doi.org/10.1109/icra.2019.8794182.

[17] F. Khan, S. Salahuddin, and H. Javidnia, “Deep Learning-Based Monocular Depth Estimation Methods—A State-of-the-Art Review,” *Sensors*, vol. 20, no. 8, p. 2272, Apr. 2020. DOI: 10.3390/s20082272. [Online]. Available: https://doi.org/10.3390/s20082272.

[18] S. Tabassum, M. S. Ullah, N. H. Al-nur, and S. Shatabda, *Poribohon-bd*, Mendeley Data, V2, 2020. DOI: 10.17632/pwyyg8zmk5.2. [Online]. Available: https://doi.org/10.17632/pwyyg8zmk5.2.

[19] N. Liu, N. Zhang, L. Shao, and J. Han, “Learning selective mutual attention and contrast for RGB-D saliency detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 9026–9042, Oct. 2021. DOI: 10.1109/tpami.2021.3122139. [Online]. Available: https://doi.org/10.1109/tpami.2021.3122139.

[20] A. Masoumian, D. G. Marei, S. Abdulwahab, J. Cristiano, D. Puig, and H. A. Rashwan, *Absolute distance prediction based on deep learning object detection and monocular depth estimation models.* Oct. 2021. DOI: 10.3233/faia210151. [Online]. Available: https://doi.org/10.3233/faia210151.

[21] R. Ranftl, A. Bochkovskiy, and V. Koltun, *Vision Transformers for dense prediction*, Mar. 2021. [Online]. Available: https://arxiv.org/abs/2103.13413v1.

[22] J. Yu and H. Choi, “YOLO MDE: Object Detection with Monocular Depth Estimation,” *Electronics*, vol. 11, no. 1, p. 76, Dec. 2021. DOI: 10.3390/electronics11010076.

[23] Z. Qu, L.-Y. Gao, S.-Y. Wang, H.-N. Yin, and T.-M. Yi, “An improved YOLOv5 method for large objects detection with multi-scale feature cross-layer fusion network,” *Image and Vision Computing*, vol. 125, p. 104 518, Jul. 2022. DOI: 10.1016/j.imavis.2022.104518. [Online]. Available: https://doi.org/10.1016/j.imavis.2022.104518.

[24] M. Ahmed, N. El-Sheimy, H. Leung, and A. Moussa, “Enhancing Object Detection in Remote Sensing: A Hybrid YOLOv7 and Transformer Approach with Automatic Model Selection,” *Remote Sensing*, vol. 16, no. 1, p. 51, Dec. 2023. DOI: 10.3390/rs16010051. [Online]. Available: https://doi.org/10.3390/rs16010051.

[25] S. A. Khan, H. J. Lee, and H. Lim, “Enhancing object detection in Self-Driving cars using a hybrid approach,” *Electronics*, vol. 12, no. 13, p. 2768, Jun. 2023. DOI: 10.3390/electronics12132768. [Online]. Available: https://doi.org/10.3390/electronics12132768.

[26] Sazed, *Vehicle Images Dataset from Bangladesh Road*, Oct. 2023. [Online]. Available: https://www.kaggle.com/datasets/sazedcse17/vehicle-bd-dataset.

[27] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, “A comprehensive review of YOLO architectures in computer vision: from YOLOV1 to YOLOV8 and YOLO-NAS,” *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, Nov. 2023. DOI: 10.3390/make5040083. [Online]. Available: https://doi.org/10.3390/make5040083.

[28] Earth Street BD, *Earth street bd - youtube channel*, https://www.youtube.com/@EarthStreetbd, Accessed: 2024-10-14, 2024.

[29] Ultralytics, *YOLOV5*, Sep. 2024. [Online]. Available: https://docs.ultralytics.com/models/yolov5/.

[30] M. Yaseen, *What is YOLOv9: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector*, Sep. 2024. [Online]. Available: https://arxiv.org/abs/2409.07813.

[31] H. Zunair, S. Khan, and A. B. Hamza, “RSUD20K: a dataset for road scene understanding in autonomous driving,” *arXiv (Cornell University)*, Jan. 2024. DOI: 10.48550/arxiv.2401.07322. [Online]. Available: https://arxiv.org/abs/2401.07322.

[32] *Monodepth2 training on KITTI dataset — gluoncv 0.11.0 documentation.* [Online]. Available: https://cv.gluon.ai/build/examples_depth/train_monodepth2.html.

[33] C. Vision and E. Z. Geometry Group, *Datasets - ETH3D.* [Online]. Available: https://www.eth3d.net/datasets.