Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

Master in Computational Data Science

Master Thesis

Intelligent Lane Detection in Autonomous Miniature Cars and Low-cost Devices

Candidate: Rachel Fanti Coelho Lima

Supervisor: Prof. Antonio Liotta

Co-Supervisor: Dr. Michele Segata

March 2023

Acknowledgements

I would like to express my sincere gratitude to the Free University of Bozen-Bolzano for providing me with the opportunity to pursue this master's degree. My deepest appreciation goes to the professors and researchers for their dedication, professionalism, and invaluable teachings. In addition, I am thankful to the staff at the Faculty of Computer Science for the guidance and support they provided; and to my classmates for their friendship and partnership.

I especially thank my supervisor Prof. Antonio Liotta and Dr. Michele Segata for proposing this study and making the autonomous cars and system available for this project. Furthermore, thanks to my friend Filipi Vaichert, for the support during the course.

I am also extremely grateful to my family, who have been a constant source of support throughout this journey. My father and grandmother (in memoriam) for their contribution to my education, both encourage me enormously in my studies and projects to these days. To my mother and my brothers, for their love, affection, and understanding, even greater in this period when my time was very limited. I cannot thank them enough for the sacrifices they made and the unwavering support they gave me. And to my son, the love of my life, who also played a huge role in motivating and encouraging me.

Abstract

1. **Motivation**: Lane detection is a critical component of autonomous vehicles, and essential for ensuring their safe and efficient navigation. Effective navigation solutions for affordable devices could contribute to the increasing use of robots and autonomous vehicles in a wide range of applications.

2. **Problem statement**: In recent years, several models have been proposed to improve the accuracy and performance of this activity, however, little is known about practical real-time implementations of these models in small-scale vehicles and low-cost devices, with limited computing and memory resources. Additionally, the suitability of deep learning models (DL) for this type of device remains unclear.

3. **Approach**: This report presents the results of a systematic mapping study aimed at gaining a deeper understanding of the models, techniques, hardware, and systems that have been employed for lane recognition on these devices. Additionally, an experimental evaluation was conducted for testing different models for lane detection in real-time on miniature cars, identifying the best-performing models.

   Three methods were analysed: (1) a model using traditional image processing techniques, (2) an end-to-end DL model proposed by Nvidia, which directly predicts angles, and (3) an ultra-fast lane detection DL model, which predicts lane boundaries. Our hypothesis is that (3) is the best-performing model due to its loss calculation formula, which is based on local and global features and could detect lanes better.

   To train the DL models, labelled data is required. In this study, an automated approach was employed for estimating the labels, which represent the steering angles or define the road markings used in the recognition process.

4. **Results**: The mapping study shows that a few models have been implemented in real-time applications for this kind of device. Among the most criticized aspects of them is the high processing and decision-making times. Another relevant point is that, although the publications provide insights into their approaches consisting of software-hardware systems, they do not allow a comparative evaluation among the models.

   The experiments conducted in this study enabled a robust and impartial comparative analysis among the methods. The results indicate that Model 1, which utilizes traditional image processing techniques, achieved a frame rate of 758 frames per second (FPS) without using any GPU, and with a memory usage of only 0.22 GB. However, it also demonstrated some scalability issues and lower accuracy compared to the other models, with a correct angle prediction rate of 86%.

   Among the DL methods, the ultra-fast model (3), which predicts lane boundaries, outperformed the end-to-end Nvidia model (2), which directly predicts steering angles. The former achieved a higher level of performance compared to the latter, with an accuracy of 95% compared to 92%. Despite its superior performance, the ultra-fast model has the disadvantage of requiring a longer processing time when run without a GPU, resulting in a lower frame rate of only 0.18 FPS. However, some adjustments were made to this model, resulting in an increase

in its frame rate to 13 FPS while maintaining the same level of accuracy. On the other hand, the end-to-end model may be simpler to implement and faster but is more prone to producing unreliable results when faced with unbalanced data.

5. **Conclusions**: In summary, the results of this study demonstrate the adequacy of DL models for addressing the lane detection recognition, even on low-cost devices, as long as minimum capacity and memory requirements are met. These models yield significantly more robust results and offer superior performance in handling variations in geometry and luminosity. A comparative analysis based on accuracy, memory consumption, and processing time is provided and serves as a useful guide for selecting the most appropriate model.

   It is essential to evaluate other traditional techniques and deep learning models. Finally, further research is needed to improve solutions, such as lighter backbones, libraries and tools, specifically tailored for mobile, embedded, and edge devices.

keywords

lane detection models, traditional image processing techniques, machine learning, deep learning, neural networks, autonomous car, miniature car, small-scale car, prototyping, raspberry pi

Contents

1 Introduction **1**

1.1 Context . . . 1

1.1.1 Autonomous vehicles and ADAS components . . . 1

1.1.2 Lane detection . . . 2

1.1.3 Miniature cars and low-cost devices . . . 2

1.2 Motivation . . . 2

1.3 Objective . . . 2

1.4 Problem statement . . . 3

1.5 Approach . . . 4

1.6 Structure of the thesis . . . 4

2 A Systematic Mapping Study **6**

2.1 Research method . . . 6

2.1.1 Research questions . . . 6

2.1.2 Study selection . . . 7

2.2 Analyses . . . 7

2.2.1 Sample characterization . . . 8

2.2.2 RQ1: What methods and techniques are used for lane detection in miniature cars or other low-cost devices? . . . 9

2.2.3 RQ2: What are the hardware and software specifications of the implemented lane detection systems? . . . 11

2.2.4 RQ3: What are the benefits of these systems? . . . 13

2.2.5 RQ4: What are the drawbacks of these systems? . . . 13

2.2.6 RQ5: How are the models for lane detection trained (e.g., computer vision, supervised or unsupervised ML)? . . . 14

2.2.7 RQ6: How is the data preparation for the experiments performed, specifically, regarding the labelling of the data (e.g., manual annotation or automatic generation of labels)? . . . 14

2.2.8 RQ7: What metrics were used for the evaluation of the lane detection systems in these devices? . . . 15

2.3 Systematic mapping considerations and conclusions . . . 16

3 Experimental Research **17**

3.1 Scope of the Study . . . 17

3.2 Selection of models . . . 18

3.2.1 The Traditional Model . . . 18

3.2.2 The End-to-end Model . . . 19

3.2.3 The Ultra-fast Model . . . 21

3.2.4 The Ultra-fast losses . . . 23

3.2.4.1 The classification loss . . . 23

3.2.4.2 The lane structural loss . . . 23

3.2.4.3 The auxiliary segmentation loss . . . 25

3.2.4.4 The overall loss . . . 25

3.3 Hypothesis formulation . . . 25

3.4 Defining the Variables of the Study . . . 26

3.5 Data collection . . . 26

3.6 Metrics . . . 28

3.6.1 Conceptual level (goals) . . . 28

3.6.2 The operational level (questions) . . . 28

3.6.3 The quantitative level (metrics) . . . 29

3.7 Development of models . . . 29

3.8 Model test environment setup . . . 29

4 Results and Discussions **31**

4.1 Challenges in Lane Detection by Model . . . 31

4.1.1 Traditional Model . . . 31
4.1.2 End-to-end Model . . . 32
4.1.3 Ultra-fast Model . . . 32
4.2 Comparison of the lane detection models . . . 34
4.2.1 Accuracy . . . 34
4.2.1.1 Visualization of the predictions . . . 34
4.2.1.2 Comparison between the deep learning models . . . 34
4.2.1.3 Comparison among all three models . . . 37
4.2.2 Memory usage and image processing time . . . 46
4.2.3 Level of satisfaction with scalability of the models . . . 49
4.2.4 Level of satisfaction with the ease of implementation of the model in code . . . 51
4.2.5 Level of satisfaction with the required effort to make the model memory-efficient and fast with no GPU . . . 51
4.2.6 Summary of the results . . . 51
4.3 Related work . . . 53
4.3.1 Related work summary . . . 53
4.3.2 Comparisons with related work . . . 54
4.4 Limitations and Threats to Validity . . . 54
4.4.1 Limitations of the research . . . 54
4.4.2 Internal validity . . . 54
4.4.3 External Validity . . . 55

5 Final Considerations **56**

5.1 Results and Conclusions . . . 56
5.2 Further Studies . . . 57

List of Tables

2.1 Searches in databases 7

2.2 Number of studies per database 8

3.1 Functions used in the Traditional Model Algorithm 20

4.1 Results of DL models, compared with the Traditional Model (ground truth) 37

4.2 Correct angle prediction rates among the different models and scenarios 37

4.3 Comparison of memory usage and image processing time among models 48

4.4 Impact of actions on image processing time, memory and accuracy 49

List of Figures

1.1 Benchmark models for lane detection on the site Papers with Code 3

1.2 Example of the labeling of road markings as ground truth 4

2.1 Characterization of the mapped studies 8

2.2 Methods and techniques used for lane detection 10

2.3 Tasks performed by ML models and outputs 11

2.4 Hardware specifications for the lane detection systems 12

2.5 Benefits and drawbacks of the software-hardware systems 13

2.6 Metrics used for the evaluation of lane detection systems 15

3.1 The self-driving robotic car and the road lane 17

3.2 Components of the miniature car 18

3.3 Model 1, the Traditional Model, which uses image processing techniques for feature extraction 19

3.4 Model 2, the End-to-end Model, a CNN architecture, which directly outputs the steering angle of the car 21

3.5 Model 3, the Ultra-fast Model, a CNN architecture, which 2 branches, one for auxiliary segmentation and another for classification 22

3.6 Grid pattern adopted in the Ultra-fast Model 22

3.7 Examples of frames captured by the cameras 26

3.8 Examples of scenarios with different conditions 27

3.9 Example of configurations for the WSL 30

4.1 Examples of segments falsely identified or undetected by the traditional image processing model 32

4.2 Distribution of angles predicted by the End-to-end Model. 33

4.3 Comparison of steering angle among different models 35

4.3 Comparison of steering angle among different models (continued) 36

4.4 Larger differences among angles 38

4.4 Larger differences among angles 39

4.5 Accuracy of steering angle predictions by the different models: a comparison of correct and incorrect predictions 40

4.5 Accuracy of steering angle predictions by the different models: a comparison of correct and incorrect predictions 41

4.5 Accuracy of steering angle predictions by the different models: a comparison of correct and incorrect predictions 42

4.5 Accuracy of steering angle predictions by the different models: a comparison of correct and incorrect predictions 43

4.5 Accuracy of steering angle predictions by the different models: a comparison of correct and incorrect predictions 44

4.5 Accuracy of steering angle predictions by the different models: a comparison of correct and incorrect predictions 45

4.6 Correct angle prediction rates among the different models and scenarios 46

4.7 Comparison of memory usage and image processing time among models 47

4.8 Prediction examples of additional Ultra-fast Model variants 50

4.9 Comparison among the performance of the three models 52

4.10 Performance comparison of Ultra-fast Models, including additional variants 52

4.11 Comparison of model satisfaction levels across scalability, ease of implementation, and memory and speed optimization for the three models 53

Chapter 1

Introduction

1.1 Context

1.1.1 Autonomous vehicles and ADAS components

Nowadays, numerous manufacturers and major technology companies are investing in and testing autonomous vehicles (AVs). A McKinsey study reveals that between 2010 and 2020 investors have spent approximately $206 billion on autonomous vehicle (AV) and smart mobility technologies [1].

The driverless car market was valued at USD 22.22 billion in 2021 according to Mordor Intelligence, considering sales of semi-autonomous and fully autonomous vehicles [2]. Research by the Boston Consulting Group simulates the effect of extensive adoption of these automobiles in five distinct types of cities, concluding that, on average, it may decline fatalities by 37% per year; reduce the need for parking areas by 35%; and diminish the energy consumption by 12% with the switch from private cars to more-efficient electric-powered AVs. Additionally, the widespread adoption of autonomous vehicles can make travel more convenient and enjoyable, while improving the efficiency of the overall transportation network and optimizing traffic flow [3].

However, critical steps need to be overcome before the adoption of completely autonomous vehicles. It is necessary to increase the reliability of current systems, develop and implement appropriate regulations, review the traffic code, and define liability for road accidents. Furthermore, the success of automated vehicle technology depends on the strategies adopted and collaboration between different disciplines and stakeholders. Questions on how to deal with the coexistence of human-driven and self-driving cars, how to deal with security issues, such as hackers and criminal actions on the system, and how to prevent significant harm to life and property must be evaluated.

While these challenges have yet to be overcome, advanced driver assistance systems (ADAS) have already been made available to consumers. These features offer several functionalities, such as lane centering, automatic lane changes, adaptive cruise control, semi-autonomous navigation for specific lanes, self-parking, and smart summon. It is estimated that in the last decade $36 billion were invested in ADAS components [1].

1.1.2 Lane detection

Lane detection and motion planning play a relevant role in the performance of ADAS functionalities, being crucial for the autonomy of these cars. It allows autonomous vehicles and robots to understand their position and road boundaries and navigate safely and efficiently. Additionally, lane detection enables autonomous vehicles and robots to identify if potential hazards, such as other vehicles or pedestrians, are on the lane and calculate the space available to define proper actions.

1.1.3 Miniature cars and low-cost devices

Miniature car prototypes have been designed and implemented in industry and research to test autonomous driving, enabling the evaluation of new products and systems under more realistic scenarios, with smaller budgets and in a safer environment. Consequently, it became necessary to test and identify appropriate lane detection models for these types of vehicles.

Moreover, the use of low-cost devices has led to an increase in the development of new technologies and applications, including those related to lane recognition. This has made it possible for more people to access these technologies, and has also led to new possibilities for innovation in this area.

1.2 Motivation

Although lane detection has been widely explored during the last three decades, there is a need to test real-time models for this activity on miniature cars and/or low-cost devices.

These models can be applied in a variety of contexts across industry, research, and education. Some examples include: in miniature vehicles for testing autonomous components and systems; in simpler autonomous vehicles such as bicycles, motorcycles, and scooters; in equipment and robots for cargo handling, inspection, or maintenance activities; in structures for traffic control systems such as traffic lights and radar, among others.

Furthermore, these systems can contribute to more affordable autonomous mobility. Simplifying the systems and hardware that make up autonomous vehicles can make them more accessible to a wider range of people.

A study developed by National Safety Council of the U.S. [4], reveals that although the availability of ADAS features in new vehicles is expanding rapidly, full penetration into the existing passenger vehicle fleet is likely to take decades. The current average age of a passenger vehicle in the United States is 11.9 years and has been growing over the years. Given the number of older vehicles on U.S. roads, it will take several years for the full safety benefits of ADAS technologies to be realized.

Finding more affordable solutions may allow faster migration to more efficient and safer transportation. At the very least they can allow in-car mobile phone or embedded devices with GPS integration for simultaneous localization and mapping, with audible alerts and warnings to drivers.

1.3 Objective

This study aims to investigate the current state of the field of lane detection models for miniature cars or low-cost devices. It will analyse practical experiments that have been conducted in the field,

including the methods, devices, hardware, systems, and metrics that have been used. The study will also test three lane detection models for use in real-time applications and evaluate their performance.

1.4 Problem statement

Most of these studies apply classical image processing methods to identify lane segments [5, 6]. However, these methods do not respond well to scenes with significant lighting and contrast variations, shadows, and obstructions [7]. Furthermore, these models rely on a combination of techniques and configurations, which vary with the diversity of scenarios, demonstrating scalability issues.

Several Deep Learning (DL) models for lane detection have been proposed to address these problems [8, 9], including architectures such as Convolutional Neural Networks (CNNs) [10, 11], Long Short Term Memory (LSTMs) [12], and generative adversarial networks (GANs) [13, 14].

New deep-learning models have increasingly been proposed to enhance lane detection accuracy and performance, as observed on the "Papers with Code" website, a free and open resource with Machine Learning papers, code, datasets, methods, and evaluation tables. Among their best-performing benchmark models at the time of this writing, are: (1) the Cross-Layer Refinement Network for Lane Detection (CLRNet), which utilizes both high and low-level features for lane detection and uses Residual Neural Network (ResNet) or Deep Layer Aggregation (DLA) as pre-trained backbones [15]; (2) the SCNN_UNet_ConvLSTM2, a hybrid spatial-temporal deep learning architecture for lane detection; (3) the CondLaneNet, a top-to-down lane detection framework based on conditional convolution [16]; and (4) YOLOPv2, a multi-task learning network to simultaneously perform the task of traffic object detection, drivable road area segmentation and lane detection [17] (figure 1.1).

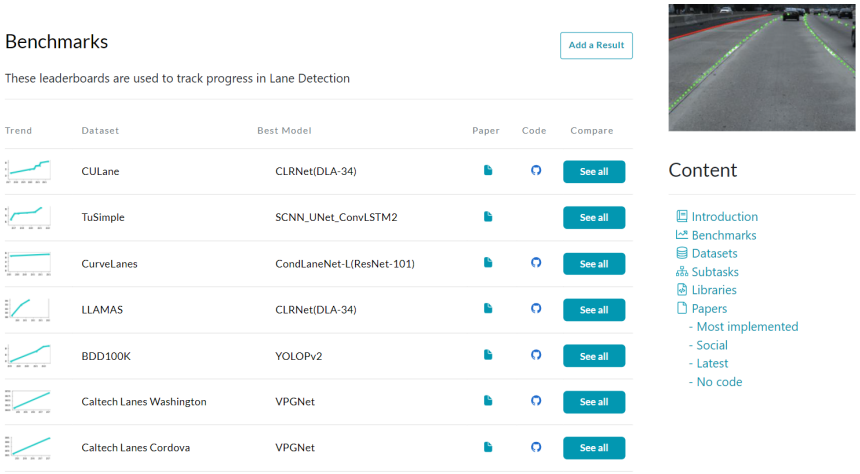


Figure 1.1: Benchmark models for lane detection on the site Papers with Code

However, usually, to apply in real-time applications, these models require to run in high-end GPU platforms. Little is known about DL lane detection algorithms in embedded systems with limited computing and memory resources. Furthermore, DL models in miniature cars can require data and labels from similar environments for training, which must be produced manually or semi-manually, a time-consuming task.

1.5 Approach

Therefore, a systematic mapping study analysing 66 documents has been conducted to better understand which models, hardware and systems are employed in real-time lane detection applications on small-scale vehicles and low-cost devices. In addition, the study can contribute to elucidating how training labels are generated, the model's outputs, and what metrics are commonly employed in their evaluation.

Moreover, this study selects and tests three existing models that adopt different methods, evaluating and comparing traditional image processing methods and DL models, identifying the best-performing models.

To train and test the machine learning models, labelled data is essential for the recognition of either the ground truth steering angles or the road markings. Figure 1.2 illustrates an example of the labeling of road markings as ground truth, showing the mapping of the x positions of the lanes to the y positions (h_samples) of the image.

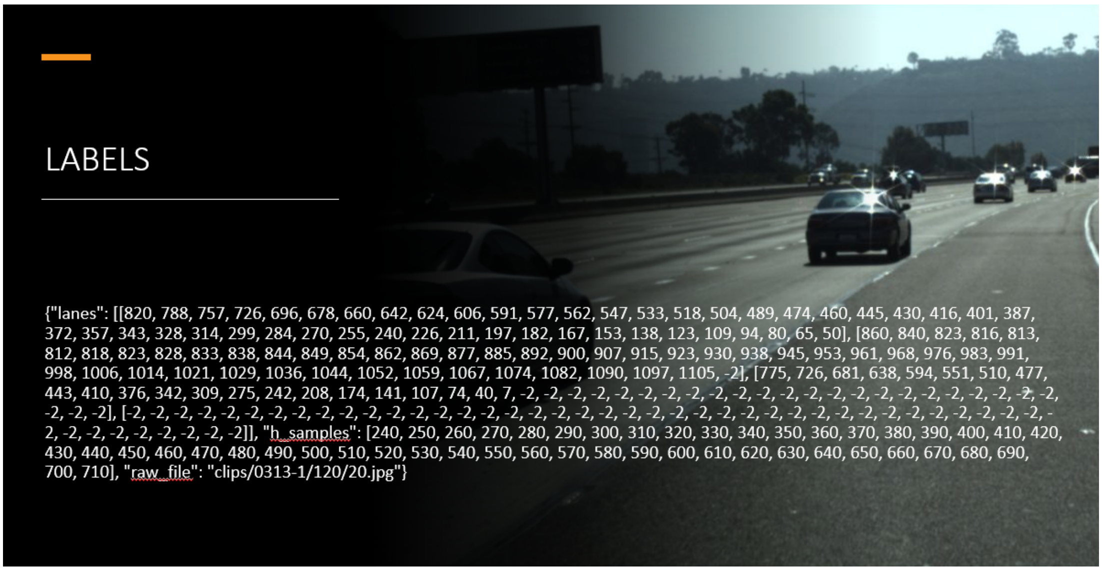


Figure 1.2: Example of the labeling of road markings as ground truth

Manually or semi-manually generating these labels can be time-consuming and labor-intensive. While there are tools and websites available to aid in the process, it still requires significant effort.

To address this challenge, this research utilized an approach for automatically estimating the labels, thereby streamlining the process and saving time by using the results of one model to train the other two deep learning models.

1.6 Structure of the thesis

This thesis is divided into five chapters: it begins with an introduction, pointing out the problems addressed by this project, its objectives, and providing an overview of the chosen approach. Then it goes over to the second chapter, which presents a systematic mapping study, providing an overview of previous work in the field and of other similar projects. The third chapter of the thesis presents a detailed description of the experimental research, including the goal and scope of the experiments, the selection of models, variables and data collection, development and testing of the models. It also

explains the methodologies used to conduct the experiments and the steps taken to ensure accuracy and reliability of the results. Chapter 4 presents the analysis of each model, detailing how the results of the first model were used to generate labels for the others. Furthermore, it contains the the main part of this thesis, which compares three different lane detection models and discusses the adjustments and revisions made to improve their performance. The result of these models are also discussed. Finally, the last section summarizes the major conclusions and presents some outlooks for further work.

Chapter 2

A Systematic Mapping Study

This chapter presents a systematic mapping study of real-time lane detection applications for small-scale vehicles and low-cost devices. The study provides an overview of the current state of research in this field by identifying and analyzing relevant publications. The first section of the chapter outlines the research questions and selection criteria used in the study, while the second section presents the results and analyses. The third section provides an overview of the systematic mapping considerations and conclusions.

2.1 Research method

2.1.1 Research questions

The goal of this systematic mapping study is to survey the literature on lane detection for miniature cars or small devices, and to identify the practical solutions and results of these implementations. The study aims to gain an understanding of the current state of research in this field and to identify the challenges and limitations related to the hardware constraints of these systems. The study will also identify trends and gaps in the current research in this field, which could be addressed by future research.

This leads to the following research questions (RQs):

- RQ1: What methods and techniques are used for lane detection in miniature cars or other low-cost devices?
- RQ2: What are the hardware and software specifications of the implemented lane detection systems?
- RQ3: What are the benefits of these systems?
- RQ4: What are the drawbacks of these systems?
- RQ5: How are the models for lane detection trained (e.g., computer vision, supervised or unsupervised ML)?
- RQ6: How is the data preparation for the experiments performed, specifically, regarding the labelling of the data (e.g., manual annotation or automatic generation of labels)?
- RQ7: What metrics were used for the evaluation of the lane detection systems in these devices?

This information from the systematic mapping study, combined with best practices from existing literature, provides valuable insights for defining the methodology and experiments to be conducted in the next chapter.

2.1.2 Study selection

Firstly, the selection of publications for this study was done by using specific keywords (table 2.1) to search for relevant publications within the title, abstract and author's keywords of documents. These searches were conducted using search engines from various sources:

- `IEEE Xplore`: https://ieeexplore.ieee.org/Xplore/home.jsp
- `ACM`: https://dl.acm.org
- `Scopus`: https://www.scopus.com/home.uri
- `ScienceDirect`: https://www.sciencedirect.com

Database	Search
ACM	(Title:((miniature OR raspberry OR prototype)) OR Abstract:((miniature OR raspberry OR prototype)) OR Keyword:((miniature OR raspberry OR prototype))) AND (Title:(lane) OR Abstract:(lane) OR Keyword:(lane)) AND (Title:(detection) OR Abstract:(detection) OR Keyword:(detection))
IEEEXplore	(miniature OR raspberry OR prototype) AND lane AND detection
ScienceDirect	(miniature OR raspberry OR prototype) AND lane AND detection
Scopus	(miniature OR raspberry OR prototype) AND lane AND detection

Table 2.1: Searches in databases

Next, the following inclusion and exclusion criteria were applied for the definition of the literature to be mapped:

- The search encompassed a variety of document types, such as conference papers, proceedings, technical reports, thesis, etc.
- The focus was on recent research, so only publications from 2018 to 2022 were considered due to the rapid evolution of computational approaches and robotics technologies.
- Publications that did not address lane detection or departures, such as those containing only traffic density or lane-changing trajectories were disregarded.
- Studies that did not take into consideration miniature cars, low-cost physical devices, or embedded systems were excluded from the study. For instance, articles that propose just algorithm prototypes.

Table 2.2 shows the number of search results per database. In the end, a total of 66 documents were analysed.

2.2 Analyses

This section presents and discusses the findings and results obtained by analyzing the data collected, addressing the research questions.

Database	Search results	Excluded publ. before 2018	Excluded duplicates	Excluded publ. without lane detection focus	Excluded publ. not focused on mini-cars or low-cost dev.
ACM	6	2	2	2	2
IEEEXplore	127	60	60	41	36
ScienceDirect	10	7	3	1	1
Scopus	226	111	54	32	27
Total	**369**	**180**	**119**	**76**	**66**

Documents published between 2018 and 2022. Retrieved: 06 August 2022.

Table 2.2: Number of studies per database

2.2.1 Sample characterization

Before delving into the specific results of the research, it is important to introduce the sample characterization. The figure 2.1 provides a summary of the characteristics of the studies consider in the analyses. It includes information on the number of publications per year, their distribution by bibliography type, and the various topics and contents covered.

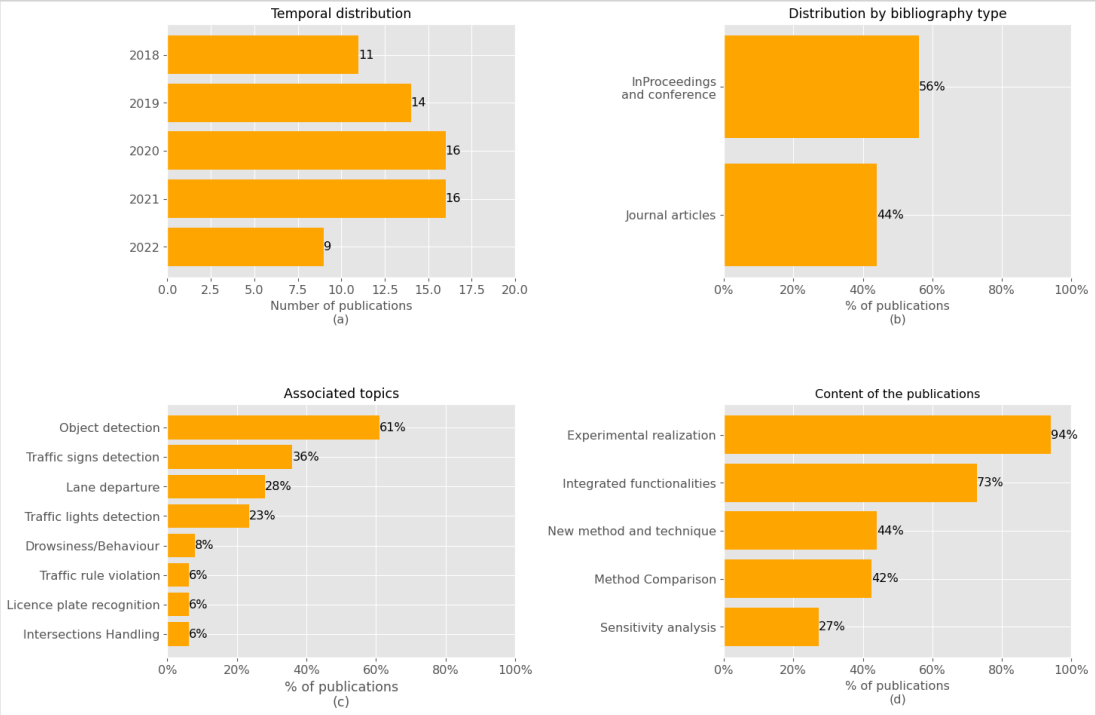


Figure 2.1: Characterization of the mapped studies

(a) Temporal distribution of the publications, (b) Distribution by bibliography type, (c) Associated topics in lane detection research (d) Content of the publication

As shown in this figure, one can observe an increasing trend in the number of publications related to lane detection over the years until 2021. The sample for this study includes conference and journal articles that cover practical implementations of lane detection. It is important to note that 94% of the documents in the sample reported included experiments performed, while the remaining 6% focused on discussing the methods for conducting experiments in this field. This highlights the growing interest and practical applications of lane detection in the field of Computer Vision.

The sample characterization also highlights that 73% of the studies analysed consider implementations with integrated functionalities. In addition to lane detection or lane departure, these documents address a range of additional topics such as object detection, traffic sign and traffic light recognition, and identification of drowsiness and abnormalities in driver behavior. Furthermore, some articles in our sample also addressed other traffic-related activities, such as the identification of traffic regulation violations and recognition of vehicle number plates or acting on traffic at road intersections. This indicates the growing importance of developing comprehensive systems for improving road safety and traffic management.

In terms of the content of the publications, approximately 44% propose new methods and techniques, while 42% make comparisons among different approaches. It is noteworthy that few publications have performed a sensitivity analysis, showing changes in system performance according to alterations in variables and parameters. This suggests that there is a need for more research in this area to better understand the robustness and generalizability of the proposed methods.

2.2.2 RQ1: What methods and techniques are used for lane detection in miniature cars or other low-cost devices?

The information from the analyses revealed that the majority of the literature analysed uses classical image processing techniques for lane detection in their experiments (80%). These methods involve a variety of feature extraction techniques to identify lane segments, as shown in figure 2.2. Among these techniques are color space conversion and intensity threshold for object selection and isolation (62%), image noise reduction and smoothness techniques (58%) like Gaussian blur (27%), and edge detection methods (58%) like Canny (47%) and Sobel (15%). Additionally, many studies also use techniques such as selecting regions of interest (ROI) (45%), histogram equalization (12%) to enhance the contrast of images, perspective transformation (24%), Hough Transform (45%) to find object instances within a certain shape class, and curve fitting and polygonal surface identification and modeling (12%). It is worth noting that some studies may have used techniques not mentioned in their articles, which therefore were not included in this analysis.

Only 23% of the publications use machine learning (ML) algorithms for lane detection, with approximately half of them using a combination of both traditional and ML methods or comparing the two techniques. Among the recommended ML methods, 87% are Convolutional Neural Networks (CNNs).

The tasks performed by these ML models are balanced between regression and classification (figure 2.3). These models generally predict the angles or directions to be taken by the car, and a small proportion (13%) predict the lane boundaries. Despite being simpler and less computationally expensive, the methods which predicts angles are limited, since they do not demarcate the available space on the track, directly defining the action. Moreover, in more complex and diverse scenarios, they may require training with a wide variety of images to be effective and correctly define the actions for the various situations.

These results indicate that although there are many deep learning (DL) lane detection models developed in the literature using elaborate techniques such as pixel-wise lane segmentation, instance-segmentation, or anchor-based tasks, when applied to practical experiments with miniature cars and low-cost devices, these models are often simplified and restricted. This suggests that while these advanced techniques may perform well in controlled lab settings, they may not be as effective when applied to real-world scenarios with limited computational resources.

It is important to note that 9% of publications have utilized sensors for lane recognition. These include photoelectric sensors, such as color and infrared (IR), as well as light amplification by stimulated emission of radiation (LASER) sensors, specifically light detection and ranging (LIDAR) sensors, often used in conjunction with global positioning systems (GPS).

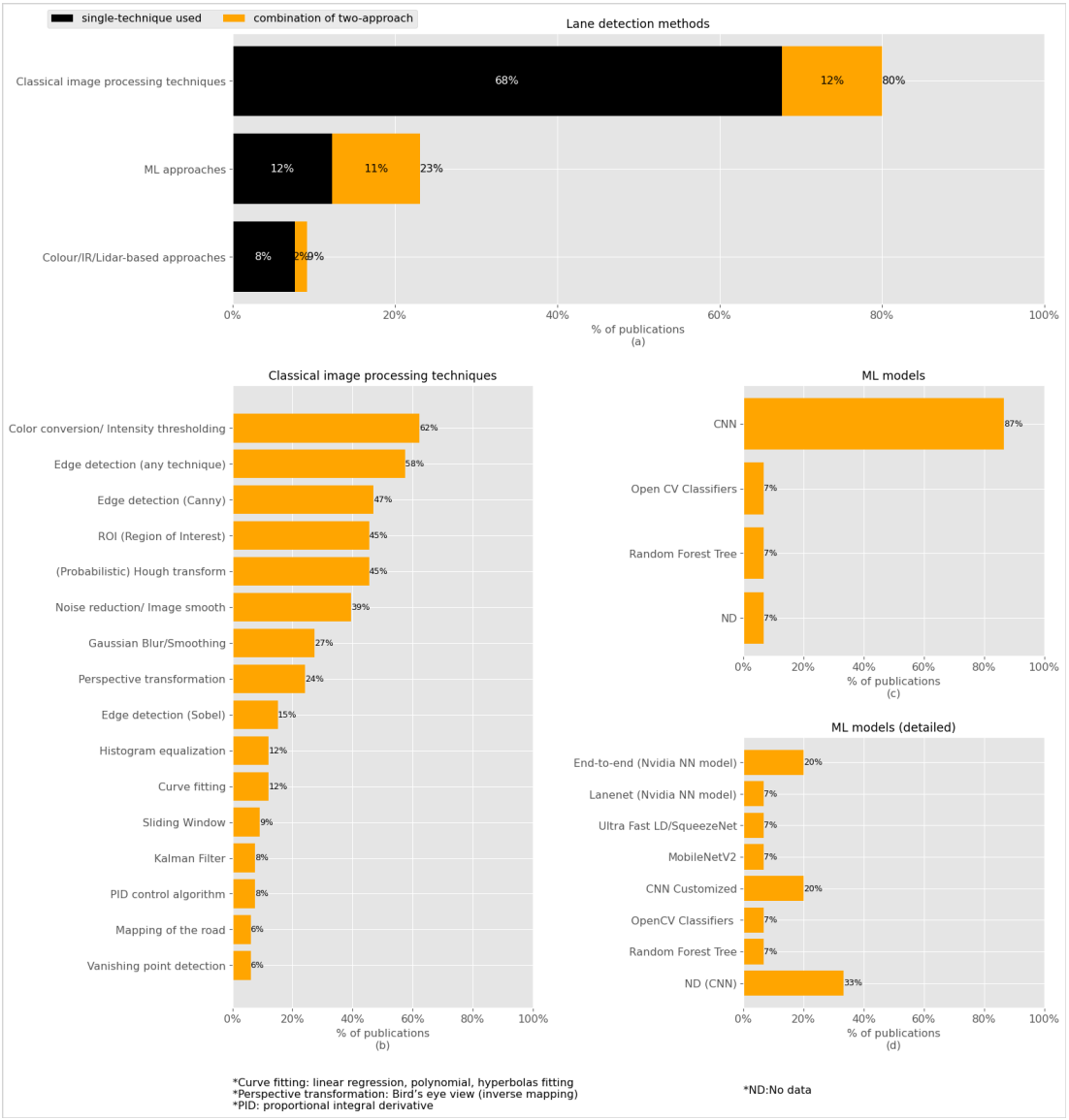


Figure 2.2: Methods and techniques used for lane detection

(a) Lane detection methods, (b) Classical image processing techniques, (c) ML models (d) ML models (detailed)

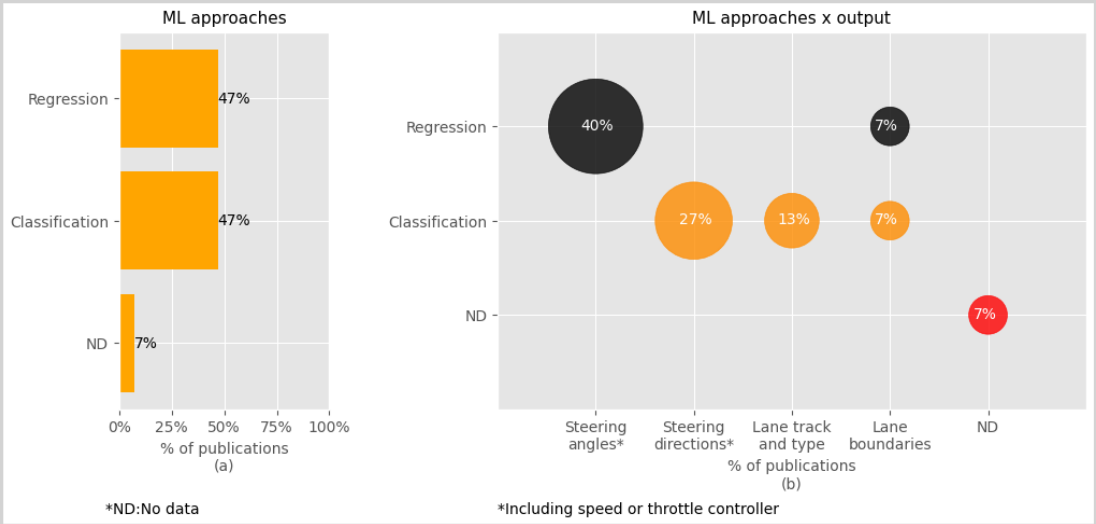


Figure 2.3: Tasks performed by ML models and outputs

(a) ML approaches, (b) ML approaches x Outputs

2.2.3 RQ2: What are the hardware and software specifications of the implemented lane detection systems?

In the systematic mapping study, 67% of the publications reported experiments with miniature cars. However, other types of low-cost devices were also considered, such as monitoring and alerting frameworks (typically embedded systems positioned inside vehicles), traffic monitoring systems (usually positioned at traffic lights), and prototype systems on mobile phones (figure 2.4).

As expected, many experiments used Raspberry as the hardware platform, as this term was included in the search for publications, along with the terms "miniature" and "prototype". Other platforms used include Nvidia Jetson (Nano or Xavier) and FPGA boards, as well as PCs, laptops, or virtual machines. Among the publications reported, 18% of them have used internal or external GPUs, with GFLOPS around 472+.

The choice of platform for the system significantly impacts the performance of models and video processing. For example, a Nvidia Jetson Nano, which is the smaller and less powerful form of the Nvidia platforms reported in publications, has 472 GFLOPS, as opposed to the 13.5 and 32 GFLOPS range obtained from the Raspberry Pi 4. However, the costs involved in acquiring the latter are much lower. Furthermore, other types of devices than computers and micro-computers have been implemented, such as embedded systems using field-programmable gate arrays (FPGAs), which are integrated circuits designed to be configured by clients using hardware-level languages.

Almost all publications (91%) have used cameras for lane detection. Cameras are well-known vision-based sensors that produce a 2D dense representation, useful for detecting shapes and colors and semantic understanding of the scene. They are efficient and widely used devices that can identify lanes and pavement markings at a low cost. However, the downside of using cameras is that they are less accurate in measuring distances than other sensors and have poor performance under extreme weather conditions.

Although most publications have used cameras for lane detection, 9% have used photoelectric sensors or laser sensors for the lane detection task, as already mentioned. In addition, a high percentage of studies report having used LIDARs and ultrasonic sensors for object detection.

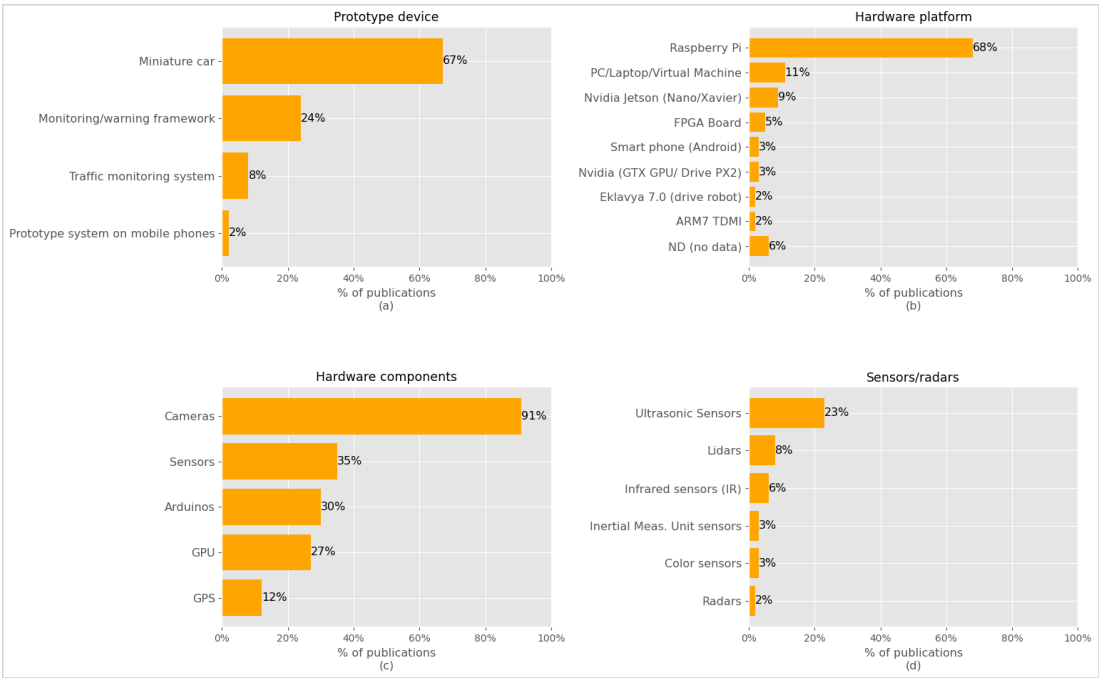


Figure 2.4: Hardware specifications for the lane detection systems

(a) Prototype device, (b) Hardware platform, (c) Hardware components, (d) Sensors/radars

Below is a brief description of each type of these sensors:

- `Photoelectric sensors` emit light from a transmitter to the surroundings, then detect the light intensity reflected from the target with a photoelectric receiver, which converts it into electrical signals. Color sensors are type of photoelectric sensors that can detect the received light intensity for red, blue, and green respectively, making it possible to determine the target color. Another example is the infrared sensor, which utilizes a detector that reacts to infrared (IR) radiation. However, photoelectric sensors are more limited to certain tasks and do not react well to gaps in lane markings, shadows, and obstructions.
- `Laser sensors`, such as LIDAR cameras, measure distances from targets by calculating the pulse laser's traveling time that they project to the target. It enables to measure distances accurately with very little information. However, the farther one object is from a sensor, the fewer points will be returned, so that more distant objects may go undetected.
- `Ultrasonic sensors` have the same working principle of the laser sensors, except for the reason that these sensors use sound waves for measuring distance, instead of laser.

These 3 types of sensors, photoelectric, laser and ultrasonic, produce point clouds, a 3D sparse representation. As drawbacks, these sensors have high cost and require computing power to provide an accurate 3d model of the environment over time. Furthermore, their use is not recommended for adverse weather conditions, such as rainy, dusty, foggy, or snowing weather.

Another interesting information is that only 12% of total publications reported have employed Global Positioning System (GPS) in their implementation for navigation and sensing purposes.

Turning now to the operating systems, Raspbian was adopted in Raspberry platforms, while Ubuntu OS and Robot Operating Systems (ROS) were employed on Nvidia and FPGA boards. Some software

simulations were occasionally employed in the experiments, such as Udacity Self-driven Car Simulator, Carla Simulator, Sensor Simulation Software-in-the-Loop (SIL), Gazebo and V-Rep.

Regarding the software libraries adopted, OpenCV stands out as the most widely used one for computer vision, being mentioned in 52% of the papers. Keras and TensorFlow were cited in 40% and 27% of the publications with ML models, respectively, while PyTorch and Torch appeared in only 7% of the papers each.

2.2.4 RQ3: What are the benefits of these systems?

The main reported strengths and drawbacks of each of these software-hardware systems approaches are depicted in figure 2.5 and briefly discussed below. It is worth commenting that this information was collected individually for each approach in the studies, and, therefore, it is not possible to make cross-comparisons among the different methods.

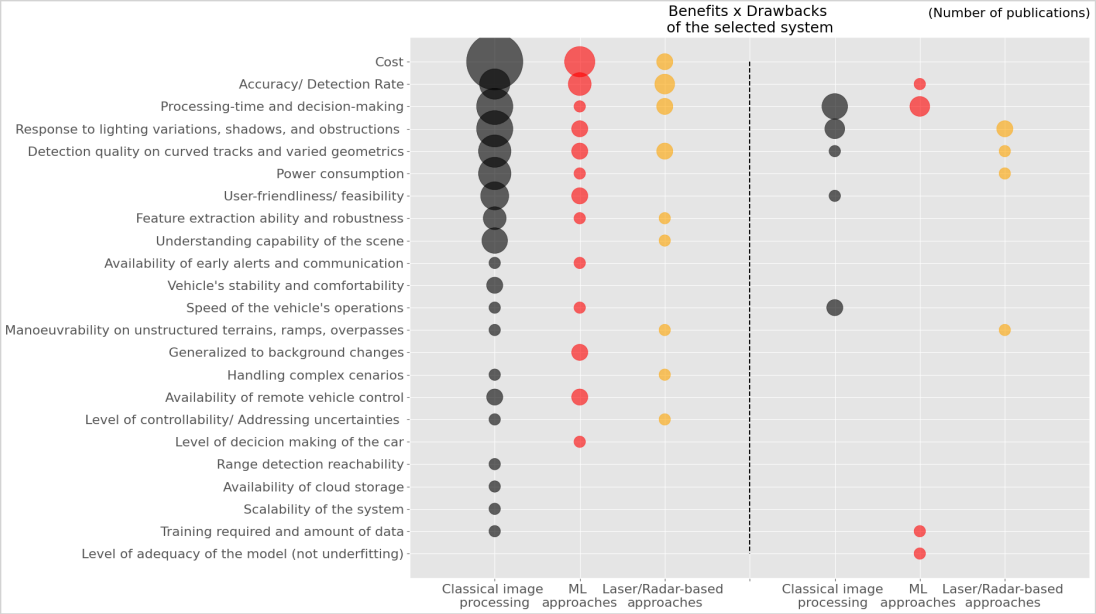


Figure 2.5: Benefits and drawbacks of the software-hardware systems

The most frequently mentioned strengths of the chosen systems are their affordability, closely followed by their exceptional accuracy in detecting lanes and fast processing time. Other benefits mentioned are the effective response of these models to lighting variations, shadows, and obstructions; the high quality in the detection of curved lanes and varied geometries; their low power consumption; the ease of use and deployment of these systems; the strong feature extraction ability and robustness; and their high scene understanding capability.

2.2.5 RQ4: What are the drawbacks of these systems?

Few publications have reported drawbacks related to the approach adopted in their studies. Among those which reported, the main critical issue informed was the high time and slight delay in the decision-making process. This issue was widely cited as a negative aspect of the adopted systems, particularly in the case of machine learning models, but also for the traditional methods. However, it was not mentioned as a drawback for the photoelectric and laser sensor-based approaches in the literature reviewed.

The low speed of the vehicle's operations was another downside identified. The speed depends on the processing time and has a high correlation with the sampling rate (frames per second) and computational overhead. The faster the vehicle is moving and the shorter is the process time, the higher risk of errors in the model's actions. Higher FPS enable reliable lane detection even at high speeds but increases the computational overhead. Some models have been criticized for not effectively balancing the trade-off between efficiency and computational overload.

Another relevant drawback identified was the difficulty of some models to respond well to scenes with significant lighting and contrast variations, shadows, and obstructions; particularly the classical image processing methods and those based on photoelectric and laser sensors. This issue was not mentioned in the experiments with ML models, which were praised for being more generalized to background changes.

A striking observation from the data analysis is that few studies make comparisons among the different models, as for example, ML models verses traditional image processing techniques, and, in addition, hardly any of these studies explicitly referred to metrics. Therefore, it was not possible to compare the performance across these models.

2.2.6 RQ5: How are the models for lane detection trained (e.g., computer vision, supervised or unsupervised ML)?

All the machine learning models used in the experiments later reported in this thesis relied on supervised learning algorithms and thus required training data with corresponding labels. In contrast, most publications involving classical image processing methods and models based on photoelectric and laser sensors did not use labels. Among these publications that did not use machine learning methods, only 26% generated labels to systematically evaluate their models and compute performance metrics.

2.2.7 RQ6: How is the data preparation for the experiments performed, specifically, regarding the labelling of the data (e.g., manual annotation or automatic generation of labels)?

To develop the models, images and/or videos were collected from databases, simulators, or captured from the real environment. When labels associated with these images were used for training and evaluating the models, they were generated through different methods:

- `Automatic annotation` refers to the process of using systems or algorithms to generate labels for images. It is often used in cases where it is possible to control the car remotely or in simulation environments, and thus the angles or directions are recorded along with the pictures.
- `Semi-automatic annotation`, in which a software tool is used to assist the human expert in the labeling process. Some publications analysed employed an annotation tool namely LabelMe, to help create labels associated with each image, or read and adjust existing ones.
- `Manual annotation`, in which a human expert manually labels the images. Especially for simple models, labels were easily generated, such as those which predict engine directions (e.g. left, right, forward) or lane numbers and types. Other models required a more refined process, such as those that design lane markings.
- `Pre-existing annotation`, refers to the use of a dataset for training a machine learning model that already includes the labels. Few publications report have been trained on this kind of labeled data.

2.2.8 RQ7: What metrics were used for the evaluation of the lane detection systems in these devices?

Analyzing the publications it was observed that little attention was paid to a consistent evaluation of the proposed lane detection methods and techniques: 35% of the publications do not perform any quantitative evaluation of their lane detection models; just 38% of them have calculated accuracy or detection rate, and even among these publications, many did it in an oversimplified way, e.g.: comparing if the number of lanes identified is the same, rather than evaluating the correct positioning and geometry of the lanes; evaluating if the directions to be taken are correct (e.g. left, right, forward) without reporting on the intensity level of the movement (figure 2.6). This lack of consistent and comprehensive evaluation makes it difficult to compare and evaluate the effectiveness of different lane detection methods and techniques.

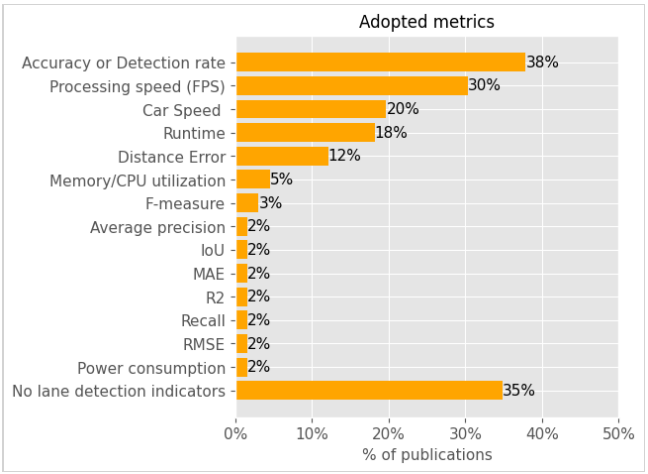


Figure 2.6: Metrics used for the evaluation of lane detection systems

Many publications in the field of lane detection do not provide sufficient details on the performance of their experiments in terms of real-time processing. Only 30% of them estimate the processing speed in terms of frames per second (FPS), and only 20% provide information on the speed of the vehicle during the experiments. Additionally, only a small number of articles that use machine learning models report processing speed (3 papers) and/or vehicle speed (2 papers). This lack of information on real-time performance makes it difficult to evaluate the practical applicability of the proposed lane detection methods and techniques. Real-time performance is essential for lane detection systems to be useful in autonomous vehicles, where the system needs to process images and make decisions in real-time.

Among the most commonly used indicators used to evaluate lane detection methods and techniques are accuracy or detection rate, average processing speed (frames per second), runtimes, and the distance (the amount of deviation) from the vehicle to the center line of the road (figure 2.6). These metrics provide information on the effectiveness and real-time performance of the proposed models. However, it should be noted that the specific metrics used and the way they are calculated can vary between publications, making it difficult to compare the results of different studies.

2.3 Systematic mapping considerations and conclusions

One of the limitations of this systematic mapping study is the keywords utilized in searching for relevant publications. Alternative keywords could be considered, as for instance, 'detect*' or 'lane departure' for lane detection, and different options for 'Raspberry'. This study presents a sample and is not exhaustive of all possible options.

Based on the analysis of the literature on lane detection for miniature cars and other low-cost devices, it can be concluded that there are still many gaps in the field.

The publications provide information on various approaches, which consist of both software and hardware components. However, they do not allow for a direct comparison between them as the studies tend to use different hardware and software configurations. To address this gap, further research is needed to conduct experiments that isolate the effects of hardware and enable a comparative study of different models.

Few publications have analysed ML models for lane detection in miniature cars and other low-cost devices. Most of the publications that apply ML methods adopt the end-to-end Nvidia Model or a similar customized CNN, which directly predict steering angles or directions. It is hardly possible to find alternative ML models for low-cost devices, especially those predicting lane boundaries.

Future research should be undertaken to investigate the suitability of these models for edge computing in miniature vehicles and low-cost devices. New methods, such as those involving pixel-wise lane segmentation, instance-segmentation, or anchor-based tasks, should be considered, and solutions to improve the real-time capability of such methods should be investigated.

The main critical issue of the approaches analysed is the high time and slight delay in the decision-making process. In future investigations, it might be possible to find an optimal solution for the trade-off between efficiency and computational overload of this models, optimizing the computation towards the fast response of the vehicle. Moreover, few publications were able to quantitatively evaluate their models and perform sensitivity analysis. A systematic evaluation of the performance of these models is necessary.

Based on the highlighted gaps, this study proposes a robust comparative analysis among lane detection models for miniature cars and low-cost devices, aiming to identify high-quality and agile decision-making solutions for use in systems with limited computational resources.

In the next chapter, we conduct three experiments to test and compare ML models and traditional image processing methods for lane detecting. The same hardware composition is used in all these experiments. Furthermore, we evaluate techniques for predicting angles and lane boundaries.

Chapter 3

Experimental Research

This chapter describes in detail the experimental research conducted as part of the study on lane detection in miniature cars and low-cost devices. The first three sections outline the scope and objectives of the experimental research, describe the lane detection models used and the reasoning behind their selection, and formulate the hypotheses. The last three sections present the variables and data, metrics, and procedures employed in the experiments to develop and test the models. This includes information on the procedures, techniques, and tools used to collect and analyze data, set up the environment, and run the models. It provides a clear understanding of the experimental design and the methods used to test the performance of the lane detection models.

3.1 Scope of the Study

The main objective of this experimental research is to compare three different real-time lane detection models for miniature cars and low-cost devices presenting their performance, with respect to their accuracy, memory consumption and processing rate.

The experiments were conducted using a miniature car on a designed track in a controlled environment (figure 3.1). The tests were performed on a laptop in a environment simulating the same characteristics of the setting in which the data were acquired.

Figure 3.1: The self-driving robotic car and the road lane

The self-driven miniature car was built by students and researchers from the Faculty of Computer Science of Free University of Bozen-Bolzano (Unibz), following the recommendations suggested by Donkey Car, an open source self-driving car platform for small scale cars. The main components of the car are presented in figure 3.2.

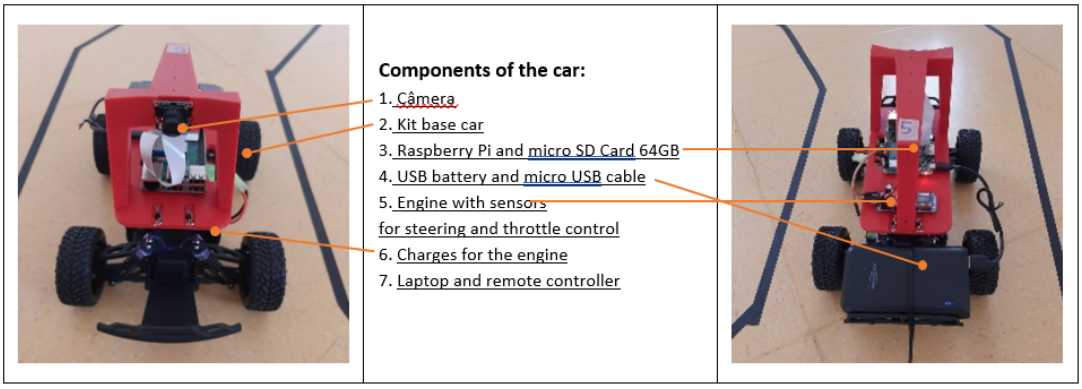


Figure 3.2: Components of the miniature car

3.2 Selection of models

This section presents the lane detection models selected for the experiment and the reasons for their choice.

Three different lane detection models were selected, namely:

- `(1) A traditional model` that uses feature extraction techniques to detect lanes on the road.
- `(2) An end-to-end model` proposed by Nvidia [10] which uses a CNN to directly predict the angle of the car.
- `(3) An ultra-fast model` [11], which uses a CNN with a Resnet 18 backbone, an auxiliary branch with a segmentation task used for training, and a module of a classification task that predicts the lane boundaries of the track.

The description of each model, with the detail of its characteristics, is presented in the following section.

3.2.1 The Traditional Model

This model employs traditional image processing techniques to recognize lanes, which are among the most widely used methods for this task, as discussed in Section 2.2.2. The reason for choosing this model is that these techniques are well-established and have been shown to be effective in lane detection.

This model employs a variety of feature extraction techniques to detect lane segments including color space conversion, image noise and detail reduction, filters, edge detection, geometric shape

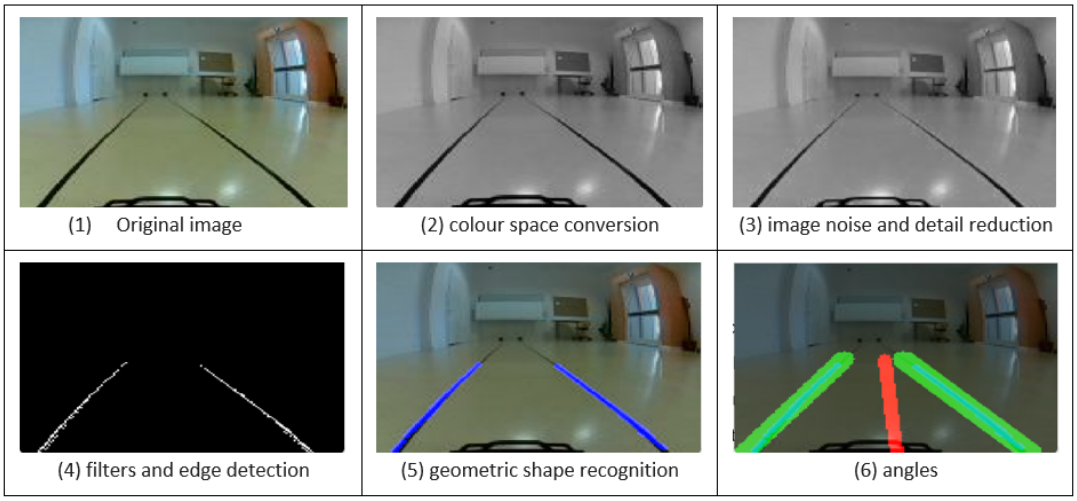


Figure 3.3: Model 1, the Traditional Model, which uses image processing techniques for feature extraction

recognition, and polygon reconstruction. Additionally, the model uses trigonometry to calculate steering angles. Photos of each step in the process were taken and are presented in figure 3.3.

To build the model, several functions were analysed to identify which ones were most effective in extracting the image features and detecting the lanes for the controlled environment of the experiment. Table 3.1 presents the most effective functions employed in each stage.

3.2.2 The End-to-end Model

This model proposed by Nvidia [10] receives an input image of 66x200 pixels and executes a regression task through a CNN, which consists of 9 layers (one normalization layer, 5 convolutional layers, and 3 fully connected layers). The first three convolutional layers use a 2×2 stride and a 5×5 kernel, while the final two convolutional layers are non-strided convolution with a 3×3 kernel size (figure 3.4).

In this model the system learns the car's steering angle directly, without the need to first detect the lane and plan the route at hand. This was one of the reasons for choosing this model. It is a straightforward solution, which streamlines the steps and maybe leads to better performance in simpler systems, which is the case. Additionally, this was the most commonly referenced machine learning model in the mapping study conducted.

To pre-process the images before running the model, the following steps were followed as suggested by [18]:

- The top half of the images were removed, as they were not relevant for lane following.
- The images were converted to the YUV colour space.
- Denoising was performed, using *Gaussian Blur*, removing noise from the data.
- Images were resized to (200, 66, 3) and normalized (divided by 255).

As the images have already been normalized in pre-processing, we disregard the first normalization layer in our model.

Stage	Function	Brief Explanation
Colour space conversion	cv2.cvtColor, cv2.COLOR_BGR2GRAY	To convert BGR images into grey.
Image noise and detail reduction	cv2.morphologyEx, cv2.MORPH_HITMISS, cv2.MORPH_DILATE	To perform advanced morphological transformations. The hit-or-miss transformation is used for shape detection of finding patterns in the image, while the dilate increases the size of foreground object.
	cv2.threshold	To apply the thresholding, given the pixel value.
	cv2.THRESH_OTSU	To automatically determine the threshold.
	cv2.adaptiveThreshold	To estimate threshold values for smaller regions and therefore estimating different threshold values for an image. This is important, since the image can have different thresholds due to variations in lighting conditions and shadowing, for example.
	cv2.ADAPTIVE_-THRESH_GAUSSIAN_C	To calculate the threshold by taking the weighted sum of the pixel values in the neighborhood area where the weights are assigned using the gaussian window technique.
Filters and edge detection	cv2.divide	To performs per-element division of two arrays, dividing the input by the image transformed.
	cv2.bitwise_not	To flips pixel values. All pixels that are greater than zero are set to zero, and all pixels that are equal to zero are set to 255.
	A customized function to define the region of interest	To filter the relevant area in the image.
Geometric shape recognition	cv2.HoughLinesP	To apply probabilistic Hough Line Transform, a more efficient implementation of the Hough Line Transform. This function returns the coordinates of the detected lines in image (x0,y0,x1,y1).
	A customized function to detect left and right segments	To combine line segments into one or two lines and classify them in left or right line.
Angles	A customized function to detect angles and avoids abrupt variations of them	To estimate the steering angles, given the lanes. If the new angle is too different from the previous, limited it to the maximum angle deviation value. In this way, the stability of the car is attempted and sudden changes due to errors are avoided.

Other methods and functions are available on the code, including cv2.GaussianBlur, cv2.equalizeHist, cv2.Canny or cv2.Laplacian.

Table 3.1: Functions used in the Traditional Model Algorithm

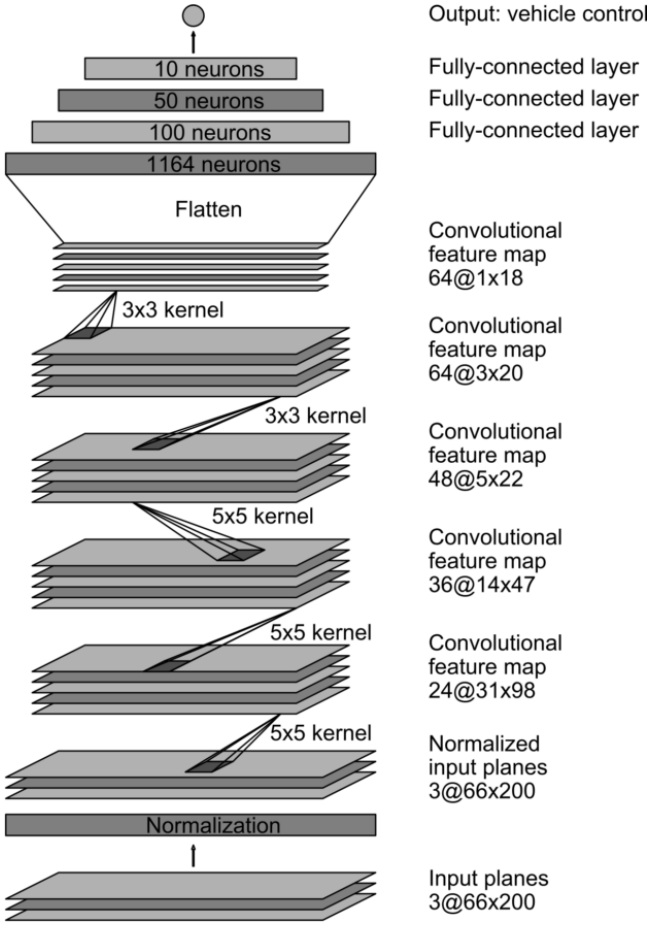


Figure 3.4: Model 2, the End-to-end Model, a CNN architecture, which directly outputs the steering angle of the car

3.2.3 The Ultra-fast Model

The ultra-fast Lane detection model [11] is a CNN model which uses Resnet 18 as the backbone. The architecture contains residual blocks with feature extractors (blue box); an auxiliary branch with a segmentation task used only for training (orange box); and a module which selects predefined cells in rows and executes a classification task (instead of segmenting each pixel of the image), and thus, reducing the computational cost (green box) (figure 3.5).

In this method, the images are arranged in a grid pattern, as shown in figure 3.6. Lanes are represented as locations on predefined rows, referred to as row anchors, along the y-axis, and in cells along the x-axis. The method predicts the probability of selecting gridding cells for the i-th lane, j-th row anchor.

To pre-process the images before running the model, the following steps were followed, as suggested by [11]:

- Images are resized to 288 pixels of height and 800 pixels of width, in RGB format.
- Transformed to tensor.
- Normalized using the mean and standard deviation of Imagenet (mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]).

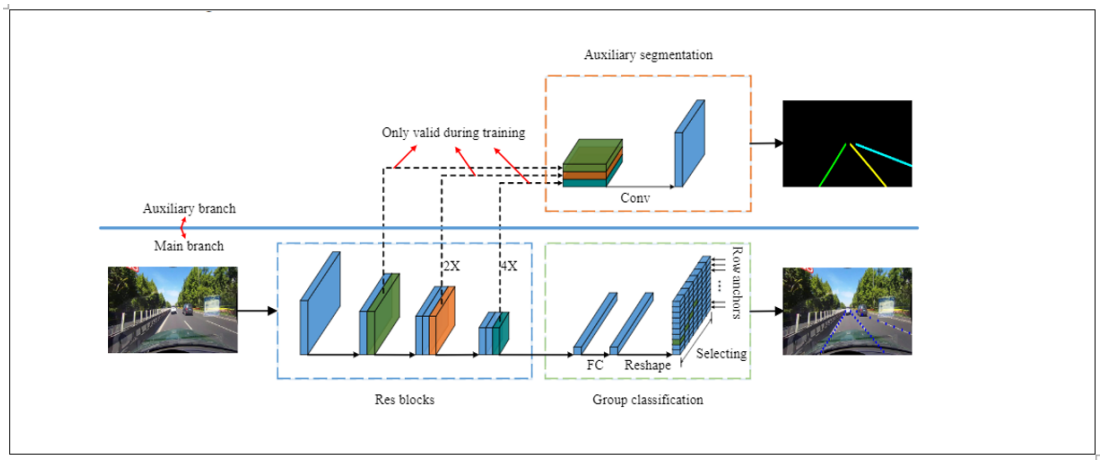


Figure 3.5: Model 3, the Ultra-fast Model, a CNN architecture, which 2 branches, one for auxiliary segmentation and another for classification

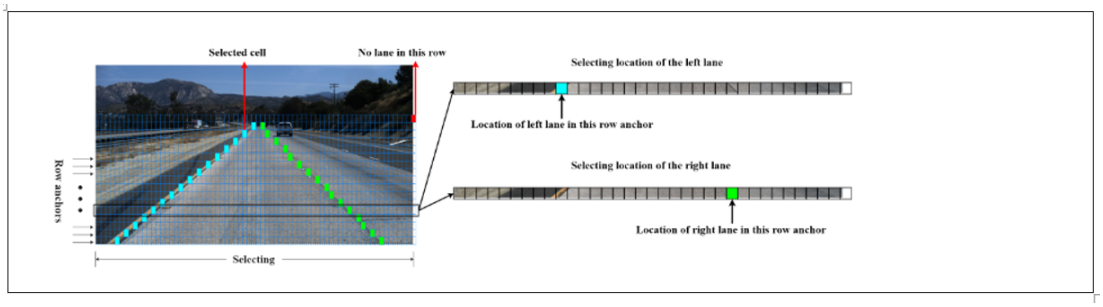


Figure 3.6: Grid pattern adopted in the Ultra-fast Model

The reason for selecting this model is its high processing speed and low computational cost, which allows it to be trained and tested on a standard machine. Moreover, it can be used in demanding scenarios such as those involving road obstructions and curved roads. This is due to the model's unique characteristics and its loss calculation formula, which incorporates both global and local features.

The local feature approach is a way to identify and extract relevant information from images focuses on their small sections. For example, for training, the model performs an auxiliary task and compute the loss formula, segmenting each pixel of the lanes.

On the other hand, another branch used for training and testing performs a classification task, applying a global feature approach. Instead of analyzing each individual pixel, this approach selects the location of the lanes on a grid, which reduces the computational cost compared to the local feature approach and uses a broader understanding of the entire image. This seeks to ensure the continuity of lanes and the definition of their shape.

The following subsection presents in detail these model losses and how they are calculated.

3.2.4 The Ultra-fast losses

The ultra-fast model considers three different types of losses:

- The classification loss.
- The lane structural loss.
- The auxiliary segmentation loss.

The first two are based in global features, while the third in local features. In the sequel, each of these losses will be presented in detail.

3.2.4.1 The classification loss

The classification loss is used to reduce the discrepancy between the output of a classification task and the true label (ground truth). Its formula is based on the cross-entropy loss as presented below:

$$L_{CLS} = \sum_{i=1}^{C} \sum_{j=1}^{h} L_{CE}(P_{i,j,:}, T_{i,j,:})$$

Where:

- C is the number of lanes
- h is the number of row anchors along the y-axis
- $P_{i,j,:}$ is a dimensional vector, representing the probability of selecting gridding cells for the i-th lane, j-th row anchor
- L_{CE} is the cross entropy loss
- $T_{i,j,:}$ is the one-hot label of correct locations

3.2.4.2 The lane structural loss

The lane structural loss consists of two loss functions which aim to model the location relations of lane points.

The first loss, named similarity loss function, aims to model the continuity of lanes by penalizing the variation of the classification vectors of lane points across adjacent rows. This loss function is based on the idea that the lane points in neighboring row anchors should be placed close to each other. The formula for this loss function is described below:

$$L_{sim} = \sum_{i=1}^{C} \sum_{j=1}^{h-1} ||P_{i,j,:} - P_{i,j+1,:}||_1$$

Where:

- C in the number of lanes
- h is the number of row anchors along the y-axis
- $P_{i,j,:}$ is a dimensional vector, representing the prediction for the i-th lane, on the j-th row anchor
- $\|\cdot\|_1$ represents L1 norm.

The second loss function, named shape loss function, is designed to model the shape of lanes. The ultra-fast method assumes that most lanes are straight, and even for curved lanes, the majority appears straight due to a perspective effect. Therefore, using the results of the classification prediction, the location of the lane on each row anchor is calculated and a second-order difference equation is used to constrain the shape of the lane. The results of the equation should be zero for straight lanes.

The formulas representing the lane location at each row anchor and the second order difference equation are presented below:

$$Loc_{i,j} = \sum_{k=1}^{w} k.Prob_{i,j,k}$$

and

$$L_{shp} = \sum_{i=1}^{C} \sum_{j=1}^{h-2} ||(Loc_{i,j} - Loc_{i,j+1}) - (Loc_{i,j+1} - Loc_{i,j+2})||_1$$

Where:

- $Prob_{i,j,k}$ is the probability of the i-th lane, the j-th row anchor, and the k-th location. It is calculated using this formula:

  $$Prob_{i,j,:} = softmax(P_{i,j,1:w}$$

  in which $P_{i,j,1:w}$ is a w-dimensional vector and $Prob_{i,j,:}$ represents the probability at each location.
- $Loc_{i,j}$ is the location of lane i on row anchor j
- w = number of griding cells along the x-axis
- C is the number of lanes
- h is the number of row anchors along the y-axis
- $\|\cdot\|_1$ represents L1 norm.

The structural loss in this model is a combination of the similarity loss and shape loss, with a loss coefficient λ to balance the contribution of the shape loss in the overall structural loss calculation.

$$L_{str} = L_{sim} + \lambda L_{shp}$$

3.2.4.3 The auxiliary segmentation loss

In the ultra-fast model, an auxiliary segmentation task is proposed to model local features using multi-scale features. The loss function used for this task is cross-entropy, and it is represented as L_{seg}, the segmentation loss. This task helps to improve the overall performance of the lane detection by incorporating local feature information in the model.

It is important to note that the auxiliary segmentation task is only used during the training phase and is removed during the testing phase. This means that the added computational cost of the segmentation task does not affect the running speed of the method during testing. This allows for the benefits of the segmentation task to be incorporated into the model without sacrificing performance during the testing.

3.2.4.4 The overall loss

In summary, the ultra-fast model recognizes the lanes by incorporating both global context and local features. The overall loss for the method is a combination of the classification loss, L_{cls}, the structural loss, L_{str}, and the auxiliary segmentation loss, L_{seg}, with the addition of two coefficients, α and β, to balance the contributions of each of loss.

$$L_{total} = L_{cls} + \alpha L_{str} + \beta L_{seg}$$

3.3 Hypothesis formulation

Three hypotheses were defined for this experiment.

The Null hypothesis:

- H0: traditional model is the one with best performance among the three.

And two alternative hypotheses:

- H1: end-to-end model is the one with best performance among the three.
- H2: ultra-fast model is the one with best performance among the three.

In this study, the best performing model is the one that attains the highest accuracy.

Our expectation is that the ultra-fast model will achieve the highest accuracy among the models tested due to its robustness and unique loss calculation formula. Unlike other models that rely only on local features, this model incorporates global features, which enables a broader understanding of the entire image.

However, it is important to consider not only the accuracy but also the video processing and memory requirements necessary to achieve the required level of FPS for a specific application. These parameters are critical factors in real-time applications, such as lane detection, and need to be carefully balanced to achieve optimal performance. Therefore, the intention of this study is to provide insights into the relationship between accuracy, memory, and FPS and help choose the best model given specific needs and constraints.

3.4 Defining the Variables of the Study

Experiments are designed to study causal relationships by manipulating independent variables and measuring their effect on dependent variables. In this study, the independent variables (inputs) are the lane detection models and the features of the images, while the dependent variable (output) is the performance.

Control variables or factors are also considered to ensure that the results of the experiment are not affected by extraneous factors. Internal factors are variables that are part of the experimental design and are being manipulated or measured as part of the experiment. These include the independent and dependent variables, and factors that the experimenter is controlling for. On the other hand, external factors refer to variables that are not part of the experimental design but may still influence the outcome of the experiment. These factors are not being manipulated or measured as part of the experiment, but they can still affect the results.

It is important to control both internal and external factors as much as possible to ensure that the effect being observed can be attributed to the manipulation or manipulation of the independent variables and not to other factors. The specific intervention, or manipulation applied to the experimental group, known as treatment, is determined by the values that the factors can take. This experimental design allows researchers to establish causal relationships between variables, and to study the effect of one variable on another.

The controlled variables of this experiment are the lane detection models, and the treatments being tested are the traditional model, the end-to-end model, and the ultra-fast model. Furthermore, in this case, the lighting conditions and the geometry of the lane are also internal factors, since, in the experiments, more pictures with different lighting variations and geometries of the lane were added to the dataset to balance the data.

3.5 Data collection

To develop model (1), pictures were captured by positioning the car in different road locations (figure 3.7). Using this model, the vehicle was driven in real settings capturing numerous pictures, of which 6.692 were selected (4.679 for training and 2,013 for tests). These pictures, in addition to the results of model (1) were then used as input and labels for predicting steering angles and lines for models (2) and (3). Images and labels that were not correctly predicted in the training of the first model were removed from the training of the other two models.

Figure 3.7: Examples of frames captured by the cameras

A diverse set of photos was collected for the experiments to evaluate the performance of the lane detection models under different conditions, as shown in figure 3.8. Photos taken in diverse lighting scenarios, such as sunlight, shadows or darkness, can help test the robustness of the models. Similarly, photos taken at various points on the track, such as on curves or straight lines, or where only one lane is visible, or no lanes are visible, can also be useful for testing the models' ability to handle variations in the scenarios.

Figure 3.8: Examples of scenarios with different conditions

The images corresponding to each scenario (a-h) were saved in separate folders, so that each variation is represented in the same proportion for the training and testing sets.

In addition to incorporating this data, which was used to ensure the model can handle various lighting conditions and lane geometries, data augmentation techniques were applied to train the machine learning models. This process generated several variations of the images, such as zoom, width and height shift, changes in brightness, and rotation. These techniques help to increase the robustness of the model and prevent overfitting.

Another important aspect of the dataset considered is the presence of similar images, especially in cases where the environment remains constant throughout the data collection process. In this particular dataset, since the car may stop at different points on the track, there may be duplicate images that provide little to no additional information for training the model.

To address the issue of redundant images, steps were taken to remove them from the dataset before splitting it randomly into training and testing sets. This approach can help to prevent overfitting, where the model becomes too specialized in recognizing specific images rather than learning generalizable features that can be applied to new, unseen data.

3.6 Metrics

The definition of the metrics used in this study was based on the "Goal-Question-Metric" approach (GQM), a method for driving goal-oriented metrics to improve and measure software quality. The GQM approach establishes a measurement model based on three levels: the conceptual level (goal), the operational level (question), and the quantitative level (metric).

3.6.1 Conceptual level (goals)

As presented in section 3.1, in this experimental research, the performance of three different real-time lane detection models for miniature cars and low-cost devices will be compared. The main goals proposed for these models are:

- `G1`: Reliability of the predictions.
- `G2`: Suitability for use in low-cost devices.
- `G3`: Speed of decision-making.
- `G4`: Scalability.
- `G5`: Ease of implementation.
- `G6`: Adaptability for use in low-cost devices.

3.6.2 The operational level (questions)

Based on the goals, five questions were formulated:

- `Q1`: How does the model affect the accuracy of lane detection?
- `Q2`: Is the model memory-efficient, making it suitable for use with low-cost devices?
- `Q3`: What is the processing time for decision making in the model?
- `Q4`: What is the level of scalability of the models?
- `Q5`: What is the level of difficulty for implementing the model in code?
- `Q6`: What is the level of effort required for adapting the model to be memory-efficient and fast?

In this study, only one question per goal was defined.

3.6.3 The quantitative level (metrics)

Quantitative or qualitative metrics have been associated with each question as follows:

- M1: Accuracy (%)
- M2: Memory consumption (GB)
- M3: Image processing rate (FPS)
- M4: Subjective rating of researcher's satisfaction related to the scalability of the models - Level (1 – 5)
- M5: Subjective rating of researcher's satisfaction related to the ease of implementing the model in code - Level (1 – 5)
- M6: Subjective rating of researcher's satisfaction related to the level of effort required for adapting the model to be memory-efficient and fast - Level (1 – 5)

The models in these experiments are evaluated primarily based on accuracy. However other multiple fronts are also considered as processing speed, memory usage, and GPU requirements. This multi-faceted evaluation approach ensures that the models are not only effective in terms of their output accuracy, but also practical and efficient in terms of resource utilization. Depending on the specific use case and application requirements, different weight may be given to each of these evaluation criteria, allowing us to select the model that best fits the needs and constraints of the given scenario.

In addition to the quantitative metrics listed above, the final three questions in the evaluation also include a subjective assessment of the researcher's satisfaction about specific topics. However, it is important to note that this information is solely for descriptive purposes and does not play a role in determining the best model. The researcher's satisfaction is the opinion of the author of this research and serves only to provide additional insight into their personal experience.

3.7 Development of models

Each model was developed using Python and trained and tested using the image sets. To ensure smooth integration with the other functionalities of the Raspberry Pi and real-time processing of individual images, a dedicated code was written for each model.

All the code for these models are available on the GitLab repository at https://gitlab.inf.unibz.it/Rachel.FantiCoelhoLima/intelligent-lane-detection-in-low-cost-devices.

3.8 Model test environment setup

To compare the models, a similar environment to that of the miniature cars was created on a Windows laptop. The Windows Subsystem for Linux (WSL) was installed to simulate the same conditions as a Raspberry Pi. This allowed the computer to run a Linux environment directly on Windows, including most command-line tools, utilities, and applications, without the need for a traditional virtual machine or dual boot setup.

The donkeycar installation files were downloaded and installed, following the recommendation of the site https://docs.donkeycar.com/, session Install Donkeycar on Windows (WSL).

In order to define settings for WSL 2, a .wslconfig file was created in the Windows user profile directory. This file allows the definition of resource allocation, such as the specification of the amount of random-access memory (RAM), swap space, number of Central Processing Unit (CPU) cores, and others. When WSL is launched, the file is read, and the system configures itself accordingly. For the model, the virtual machine (VM) memory was limited to 3 GB (figure 3.9, and the VM was set to use 4 virtual processors.

```
rachelfanti@DESKTOP-ED3T7OL:~$ free -h --giga
               total        used        free      shared  buff/cache   available
Mem:            1.9G        277M        1.5G        2.0M        173M        1.5G
Swap:           1.0G          0B        1.0G
rachelfanti@DESKTOP-ED3T7OL:~$
```

Figure 3.9: Example of configurations for the WSL

In the following section, the performance analysis and results for each model are presented, along with an examination of any challenges encountered during their development, as well as their strengths and limitations. Additionally, the process of generating the labels will also be discussed in this chapter, as it was generated using the results from the first model.

Chapter 4

Results and Discussions

This chapter presents an analysis of the performance of lane detection models, as well as the main findings and discussions.

The first section of this chapter presents an in-depth analysis of the performance of lane detection models, including the approach used for generating labels in an automated, yet approximate manner. It delves into the various challenges faced during the lane detection process and possible solutions to address them. In the second section, the performance of the three lane detection models is compared, and the adjustments and revisions made to improve their performance are discussed.

4.1 Challenges in Lane Detection by Model

In this section, the key challenges faced by each model are highlighted and illustrated. It explores the difficulties encountered during the process of detecting lanes and presents the solutions adopted to overcome them.

4.1.1 Traditional Model

The traditional image processing techniques applied in this model accurately identified most of the line segments. Nevertheless, some segments were falsely identified due to lighting effects or due to the presence of other lines on the scene. Other segments went undetected, as illustrated in figure 4.1.

The false segments present in the scene were partially eliminated by defining the Region of Interest (ROI). The left and right segments related to the lower half of the image were then distinguished, and weights were assigned to each segment based on their length, with longer segments receiving a higher weight. These segments were then combined into a single left line and a single right line for guiding the car (green lines).

These resulting left and right lines were utilized to determine the steering angles for the car in this model and served as labels for training the Deep Learning models.

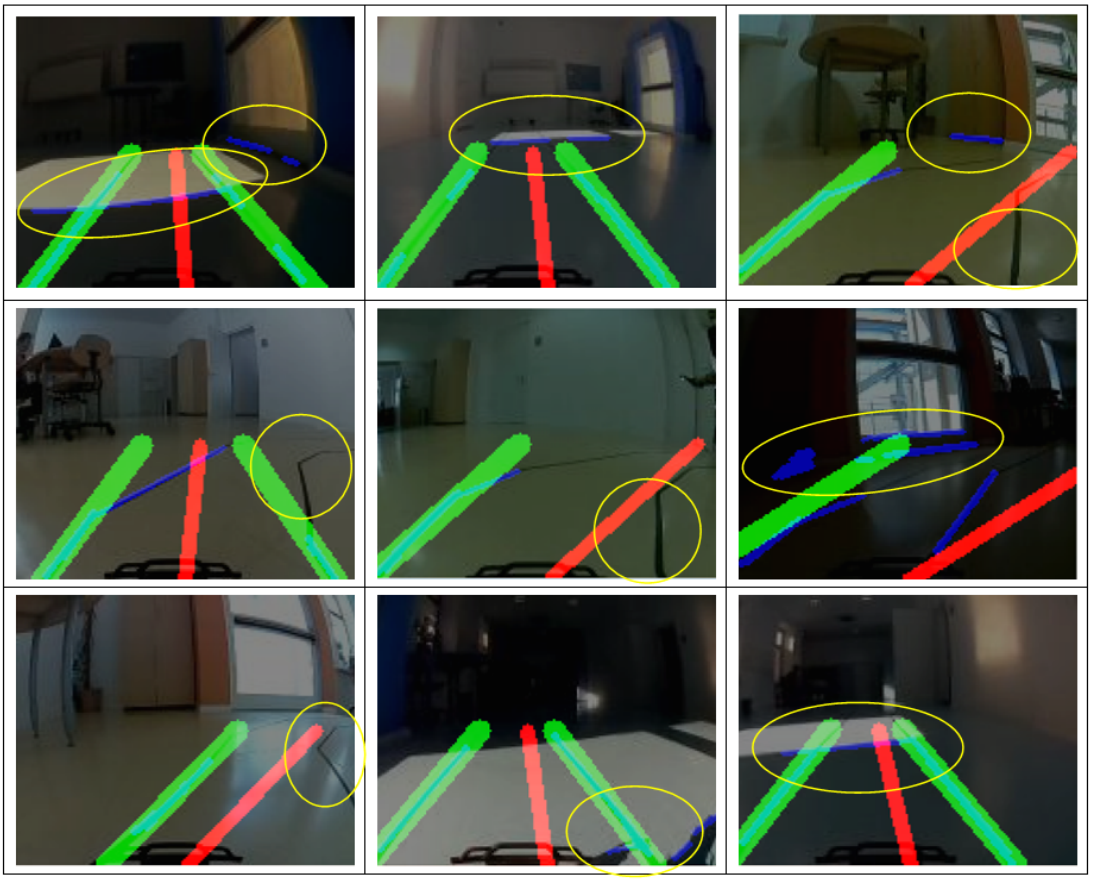


Figure 4.1: Examples of segments falsely identified or undetected by the traditional image processing model

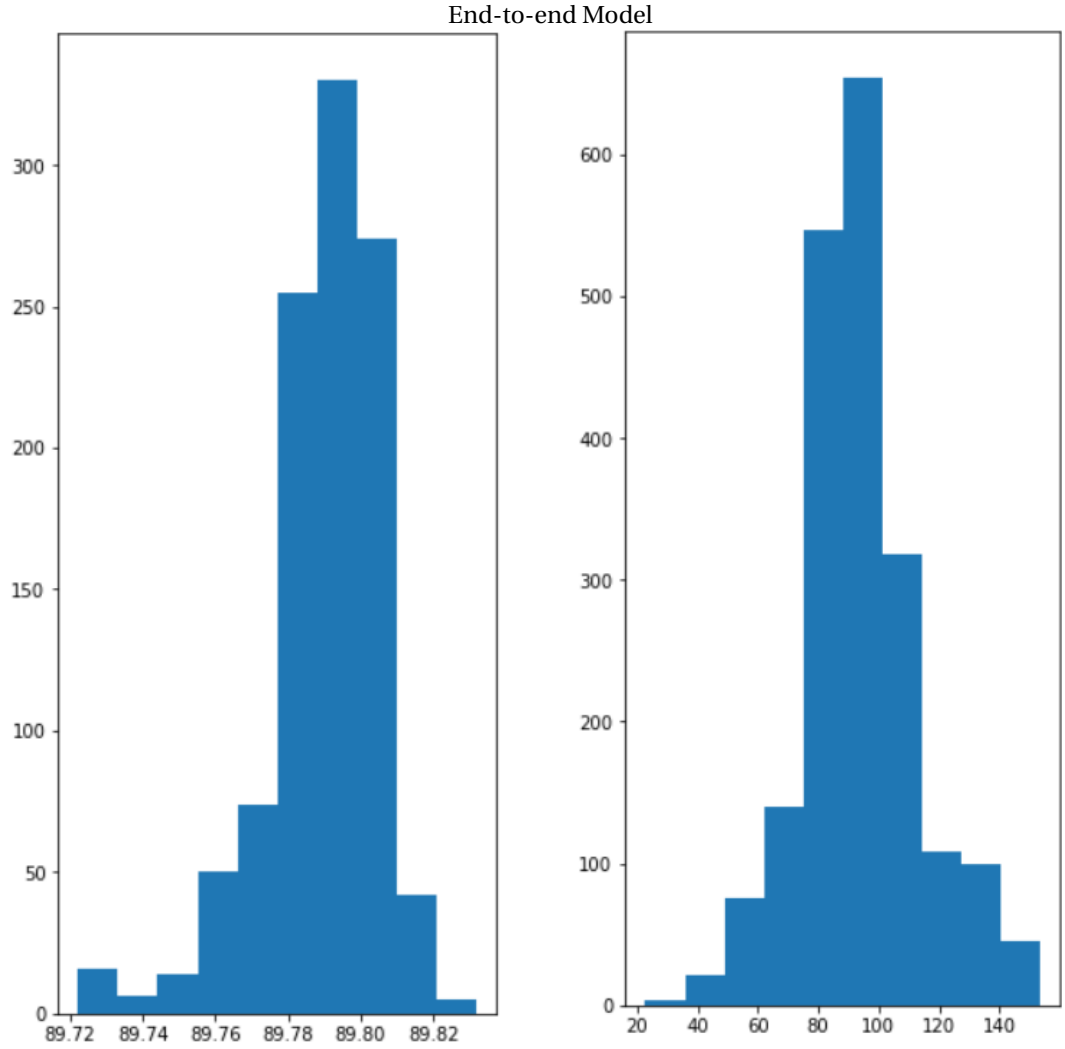
4.1.2 End-to-end Model

This model was found to exhibit high sensitivity to the proportion and characteristics of the input images. For instance, when initially trained using a set of consecutively captured images while driving the car, the model primarily predicted almost 90-degree angles due to a majority of the images containing straight lines. However, when using data in which most of the images contained curves, the model's predictions leaned towards either no straight lines or 90-degree angles. Balancing the data for training the model to perform well in various scenarios was a major concern (figure 4.2).

4.1.3 Ultra-fast Model

The Ultra-fast Model yields a more complex algorithm than the others and it functionally depends on several libraries. Additionally, it demands a more involved configuration setup and conversion of data, as for example, in the training process where the labels generated from the Traditional Model must be converted to a grid format by defining the x values for the established y values for each lane on the track.

The model performed well with minimal adjustments, however, when tested on the Linux environment created, similar to the environment on the Raspberry Pi, it was necessary to simplify the code for use on systems with limited GPU and memory.



(a) First attempt - original distribution of the images (scale from 89.7 to 89.9).

(b) Last attempt - enhanced distribution of images with a wider range of track geometries and lighting conditions (scale from 20 to 160).

Figure 4.2: Distribution of angles predicted by the End-to-end Model.

In this case, the simplification strategies implemented include streamlining the code, optimizing functions, and reducing options and configuration parameters. The *argparse* function - parser for command-line options, arguments and sub-commands - was also eliminated due to its impact in processing time. To further improve performance, the memory usage and time spent in each step of the model were analysed.

Despite the optimization efforts, the need for an additional GPU for real-time prediction of steering angles and execution of actions still persists. This will be further discussed in the following session.

4.2 Comparison of the lane detection models

Adhering to the GQM methodology outlined in the section 3.6, a comparison of the models will be presented.

A significant emphasis was placed on the analysis of the accuracy because it is a crucial factor in determining the effectiveness and reliability of the models, directly impacting their ability to perform the intended task and deliver accurate results. Hence, it is important to evaluate and compare the accuracy of the models in a thorough and systematic manner.

4.2.1 Accuracy

4.2.1.1 Visualization of the predictions

For each model, lanes and angles were predicted, as shown in the figure 4.3.

4.2.1.2 Comparison between the deep learning models

The deep learning models were analysed and compared with the labels generated by the Traditional Model, used as as ground truth.

As previously stated in Section 3.5, images and labels that were not accurately predicted in the training of the first model were eliminated from the training of the subsequent two models. Additionally, to calculate the results of deep learning on the test set, images and labels that were not accurately predicted in the testing of the first model were also removed.

For a thorough and well-rounded assessment of the model's reliability, the following metrics were analysed:

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE).
- Mean Absolute Error (MAE)
- Accuracy (considering an angle threshold equal to 10 degrees).

Additional metrics to the accuracy were considered to complement and avoid mono-method bias.

In table 4.1, the results for the test of the deep learning models are presented. Furthermore, the testing subsets 1 and 2 are included as they reflect the appropriate distribution of images when the car is in operation. The remaining sets in the database include additional images depicting various lane geometries and lighting conditions, which alter the distribution and are not representative of normal operation, but are considered in the evaluation of the complete database.

It is evident that the Ultra-fast Model outperforms the End-to-end Model. Nevertheless, the End-to-end Model still achieved satisfactory results. Additionally, it is apparent that for the subsets, the results of both models improve even further.

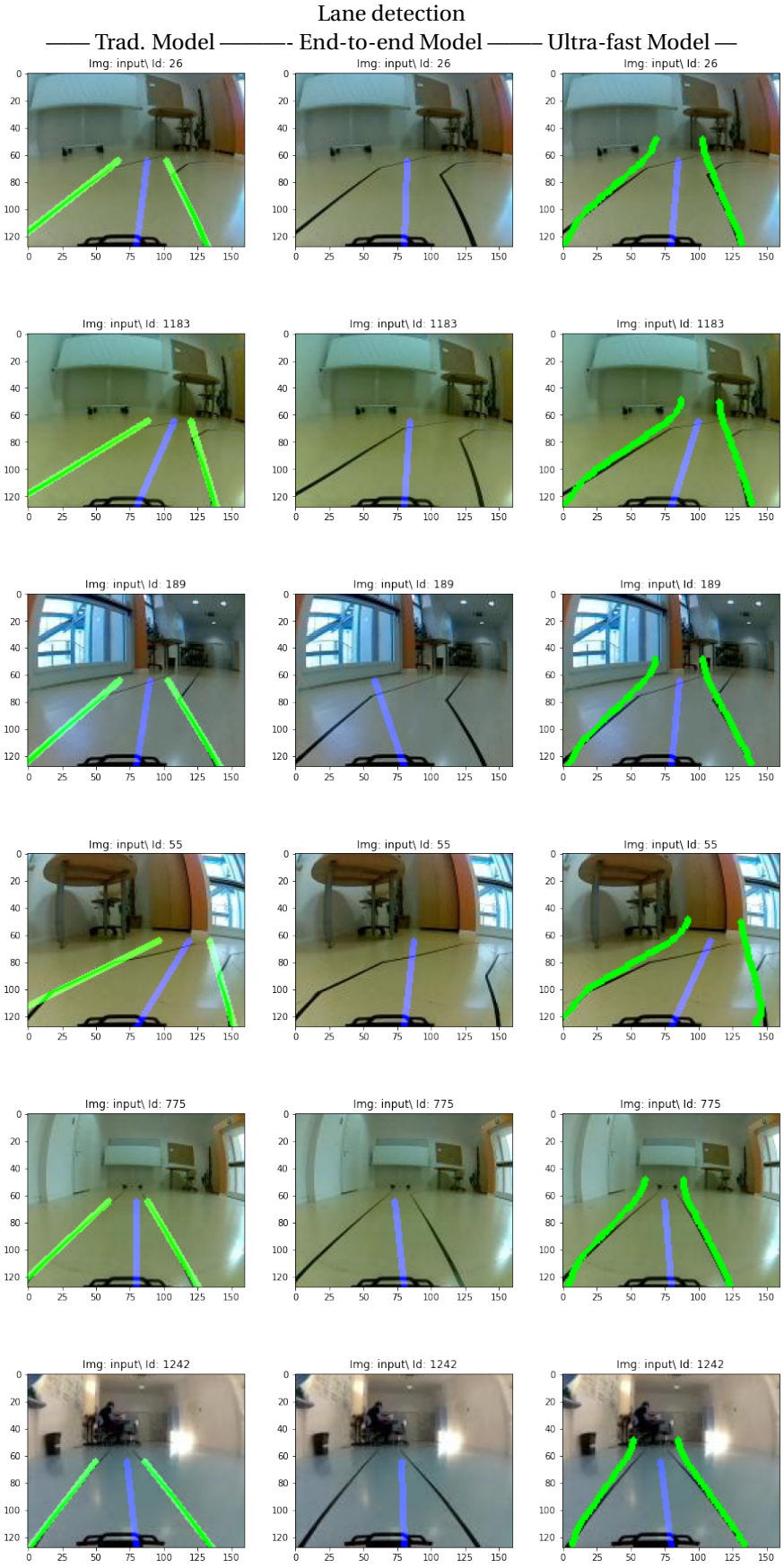


Figure 4.3: Comparison of steering angle among different models

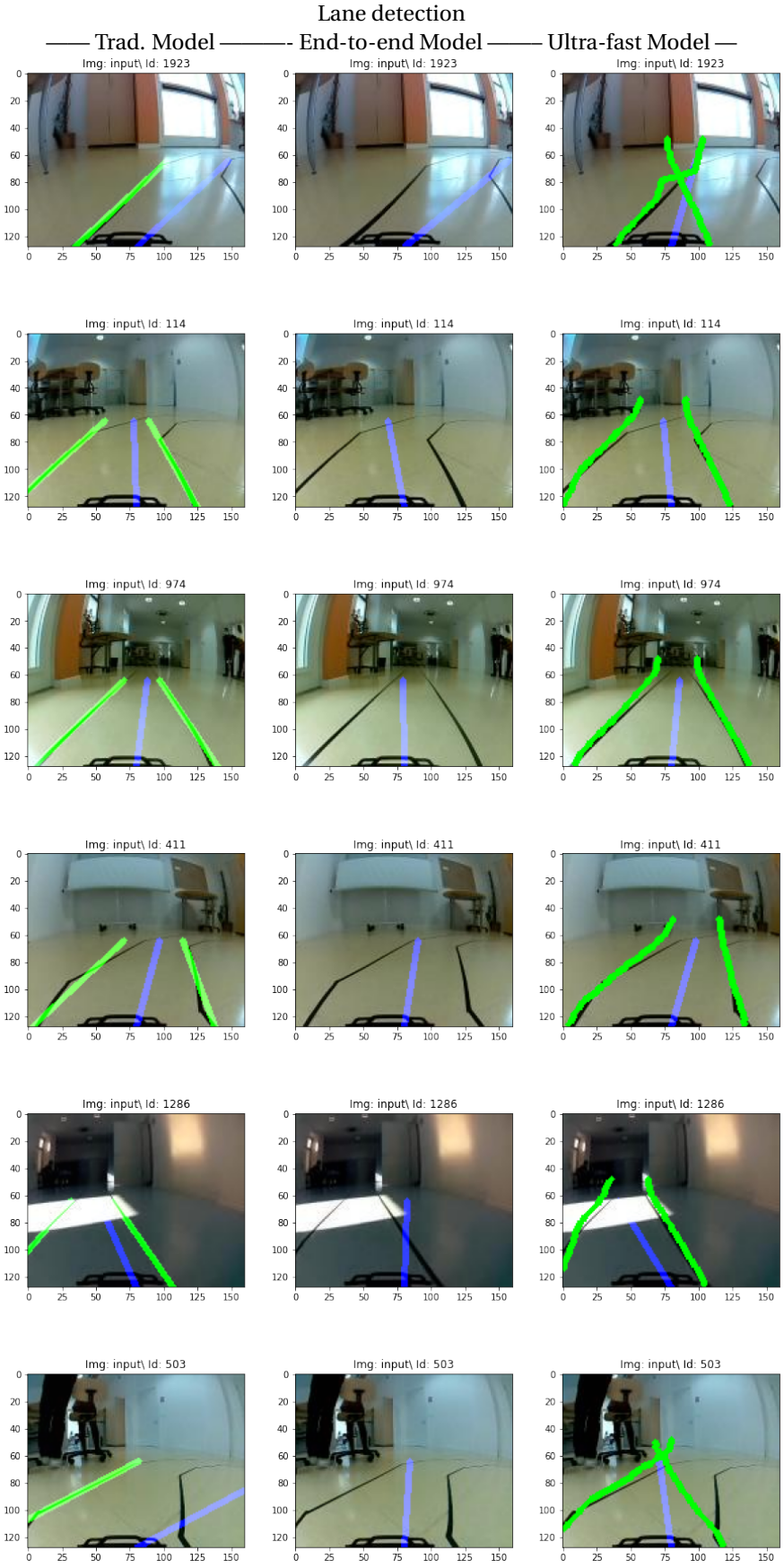


Figure 4.3: Comparison of steering angle among different models (continued)

Models		End-to-end				Ultra-fast			
Set		MSE	RMSE	MAE	Acc	MSE	RMSE	MAE	Acc
Train	All database*	350	19	12	62.42%	190	14	6.9	85.64%
Test	Subset 1	200	14	8.6	74.80%	240	4.9	3.3	95.40%
	Subset 2	190	14	8.4	76.37%	260	5.1	3.2	98.01%
	All database*	320	18	12	61.76%	160	13	6.6	85.67%

Table 4.1: Results of DL models, compared with the Traditional Model (ground truth)

4.2.1.3 Comparison among all three models

Observing the largest differences in the angles between the Ultra-fast Model and the traditional one, it can be noted that the main gaps occurred during instances when the Traditional Model did not perform well (figure 4.4).

Since the machine learning models were developed based on the Traditional Model, there was no ground truth available to compare their performance with the Traditional Model.

Therefore, all the images and predicted steering angles in the test set were subjected to visual inspection. To eliminate potential biases in the research related to the models, a well-defined criterion was established to determine whether the angle was accurately predicted or not.

A line was drawn at one-third of the height of each image, and it was determined whether the car would stay on the track or not. Even if the prediction was not perfect, it was assumed that the car would have time to adjust its direction in the next frame.

The figures 4.5 show examples of images that were evaluated as accurately predicted and inaccurately predicted for each model.

The table 4.2 summarizes the results of the accuracy of the three models based on their ability to correctly predict the angle.

Group	Scenarios	Nº Pict.	Nº Pict. Correct Angle			% Pict. Correct Angle		
			Trad.	End	Ultra	Trad.	End	Ultra
(1)	Basic 1	680	631	658	680	93%	97%	100%
(2)	Basic 2	530	404	507	530	76%	96%	100%
(a)	Sunlight incidence	94	94	83	94	100%	88%	100%
(b)	Lighting variance	84	60	76	84	71%	90%	100%
(c)	Darkness	150	134	131	147	89%	87%	98%
(d)	Curves	300	250	257	266	83%	86%	89%
(e)	Diagonal straight	23	18	22	23	78%	96%	100%
(f)	Car off centre line	120	118	82	119	98%	68%	99%
(g)	One line visible	20	12	8	14	60%	40%	70%
		2013	**1731**	**1834**	**1969**	**86%**	**92%**	**95%**

Table 4.2: Correct angle prediction rates among the different models and scenarios

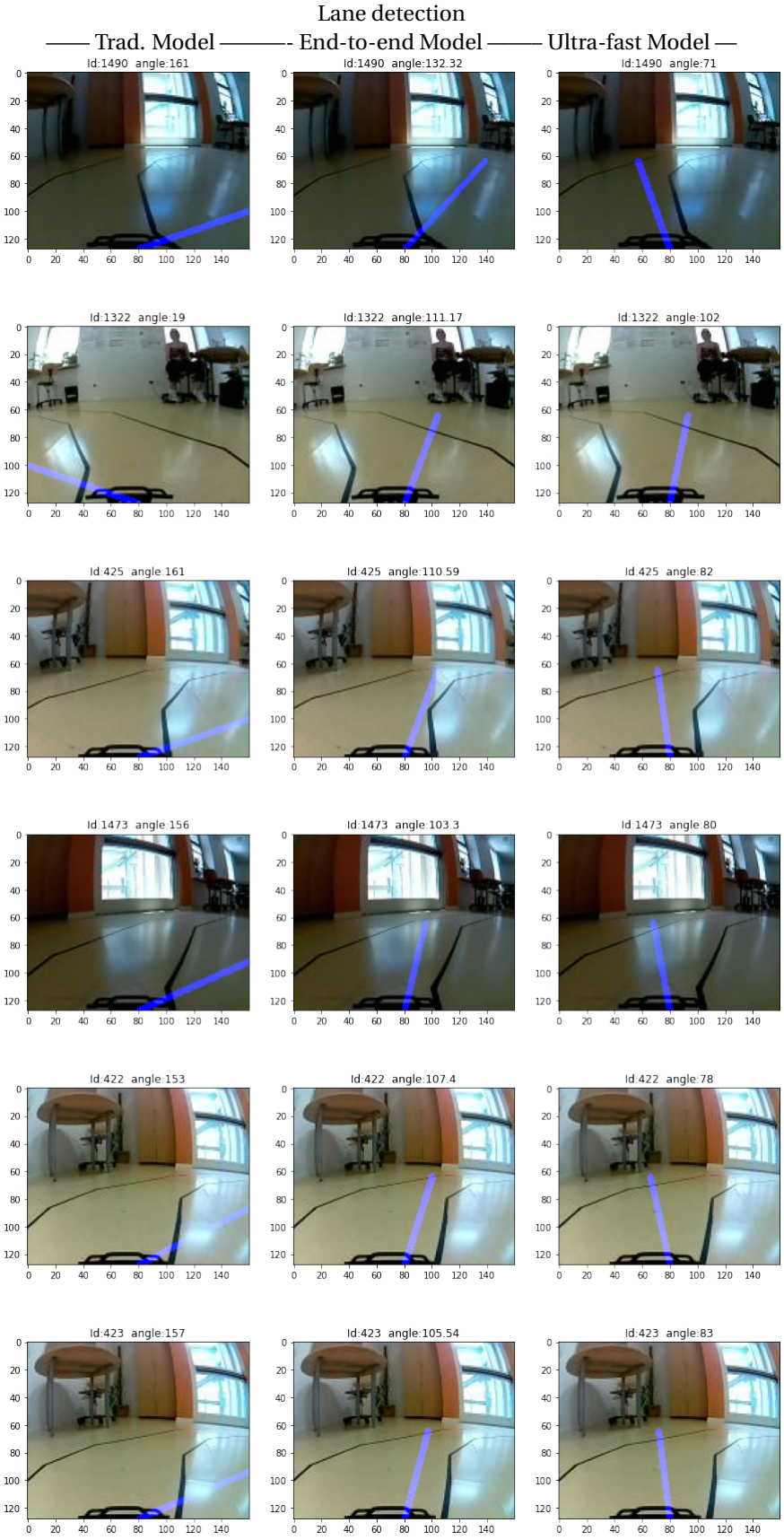


Figure 4.4: Larger differences among angles

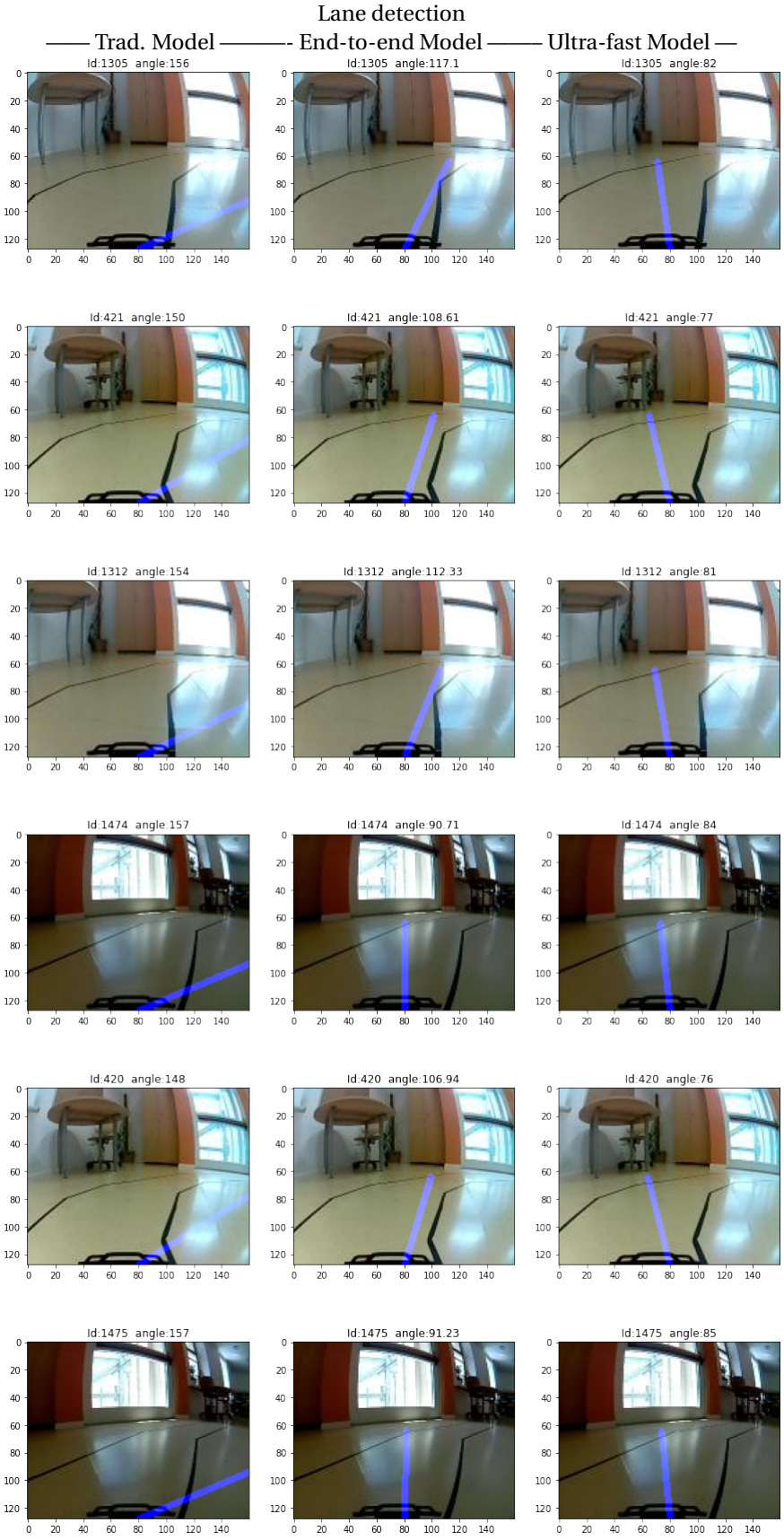


Figure 4.4: Larger differences among angles

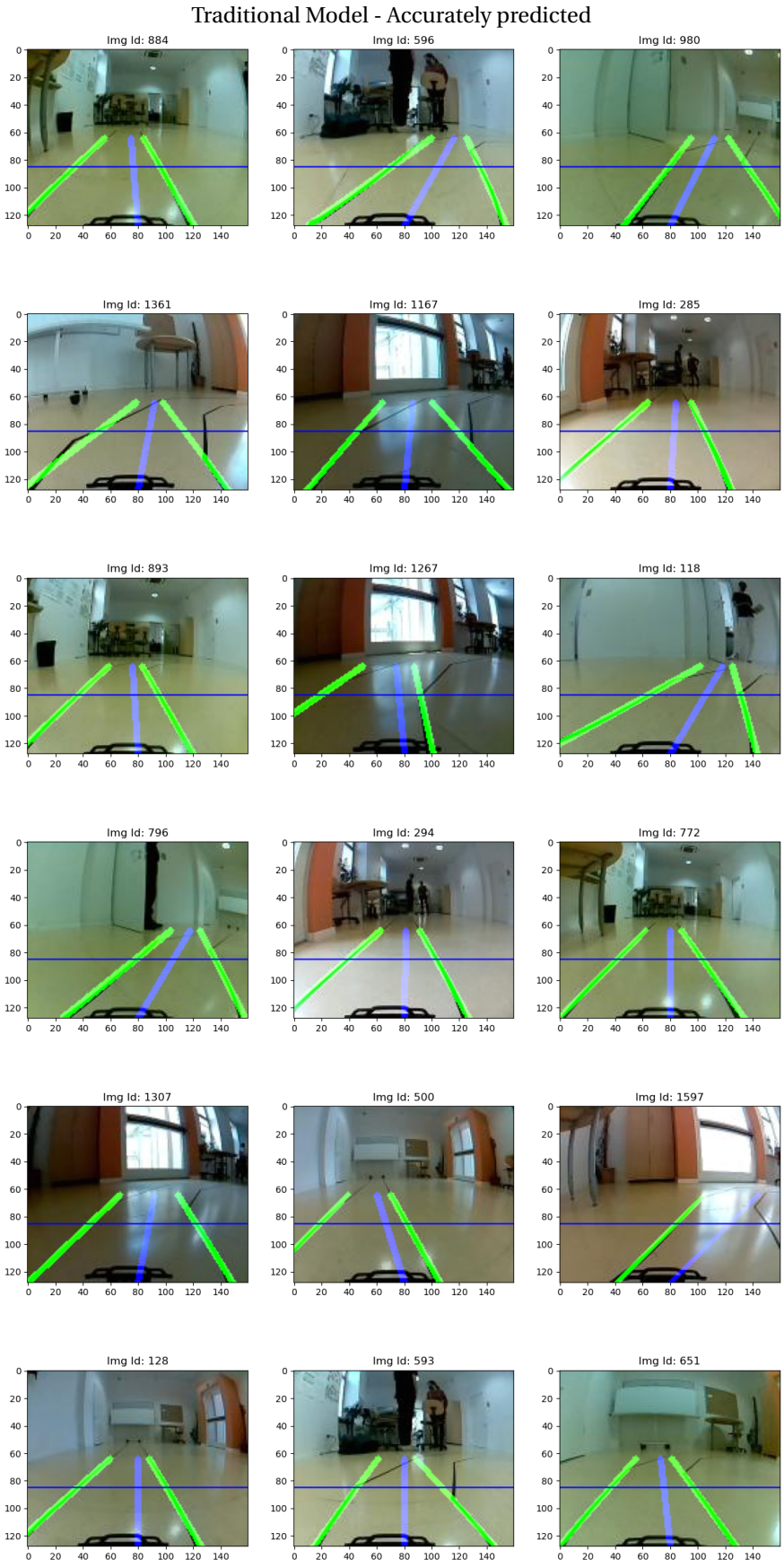


Figure 4.5: Accuracy of steering angle predictions by the different models: a comparison of correct and incorrect predictions

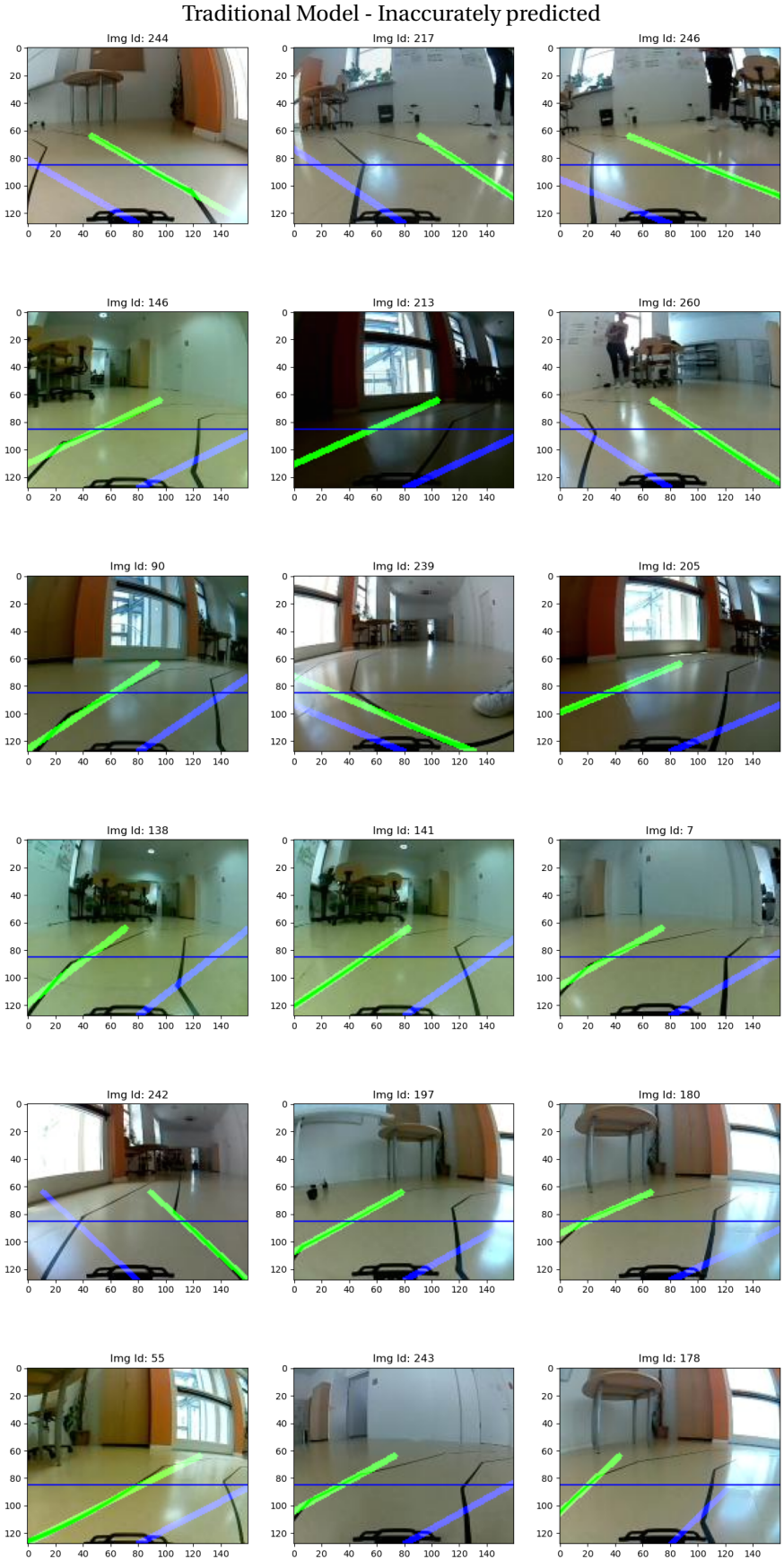


Figure 4.5: Accuracy of steering angle predictions by the different models: a comparison of correct and incorrect predictions

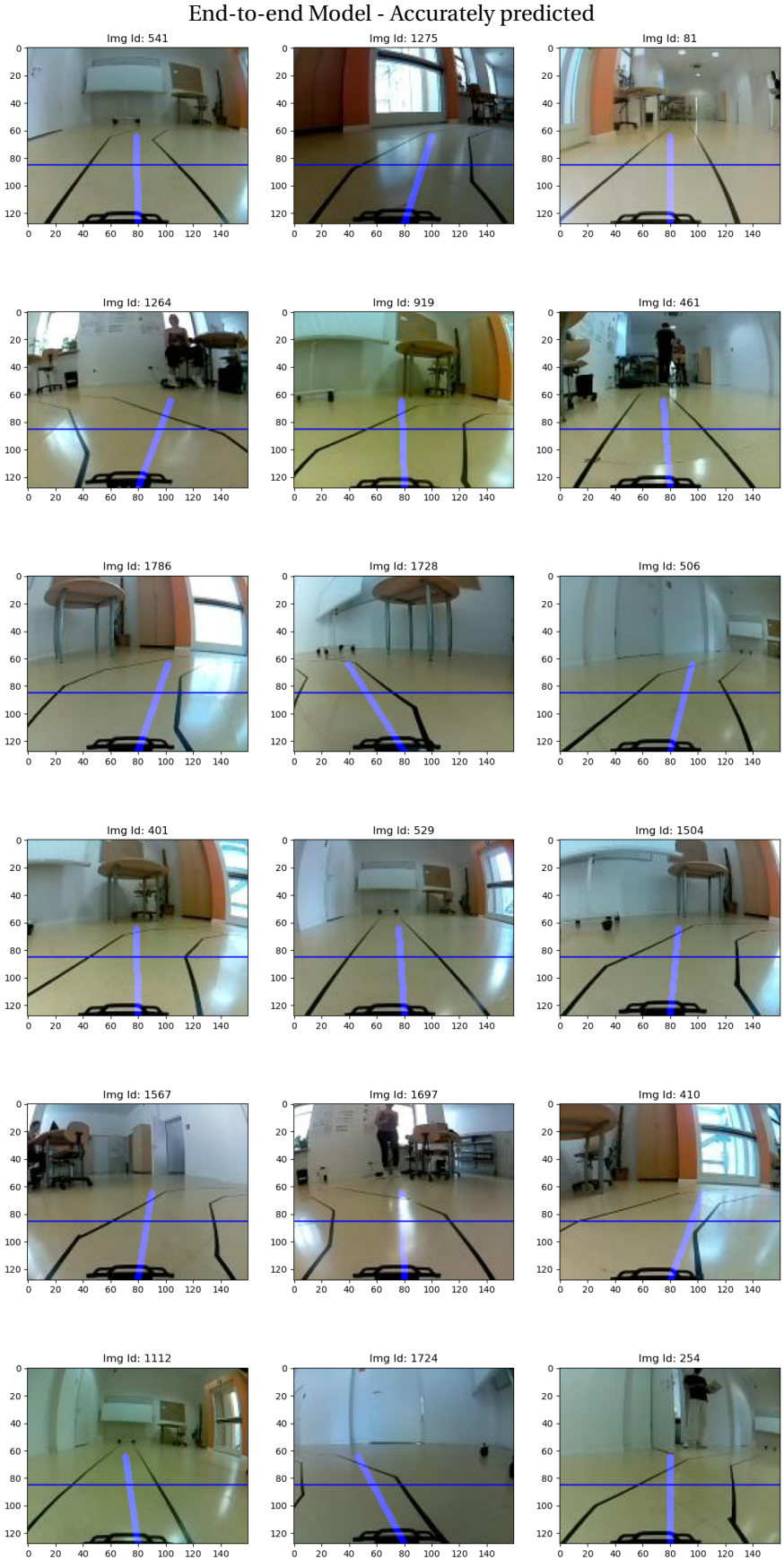


Figure 4.5: Accuracy of steering angle predictions by the different models: a comparison of correct and incorrect predictions

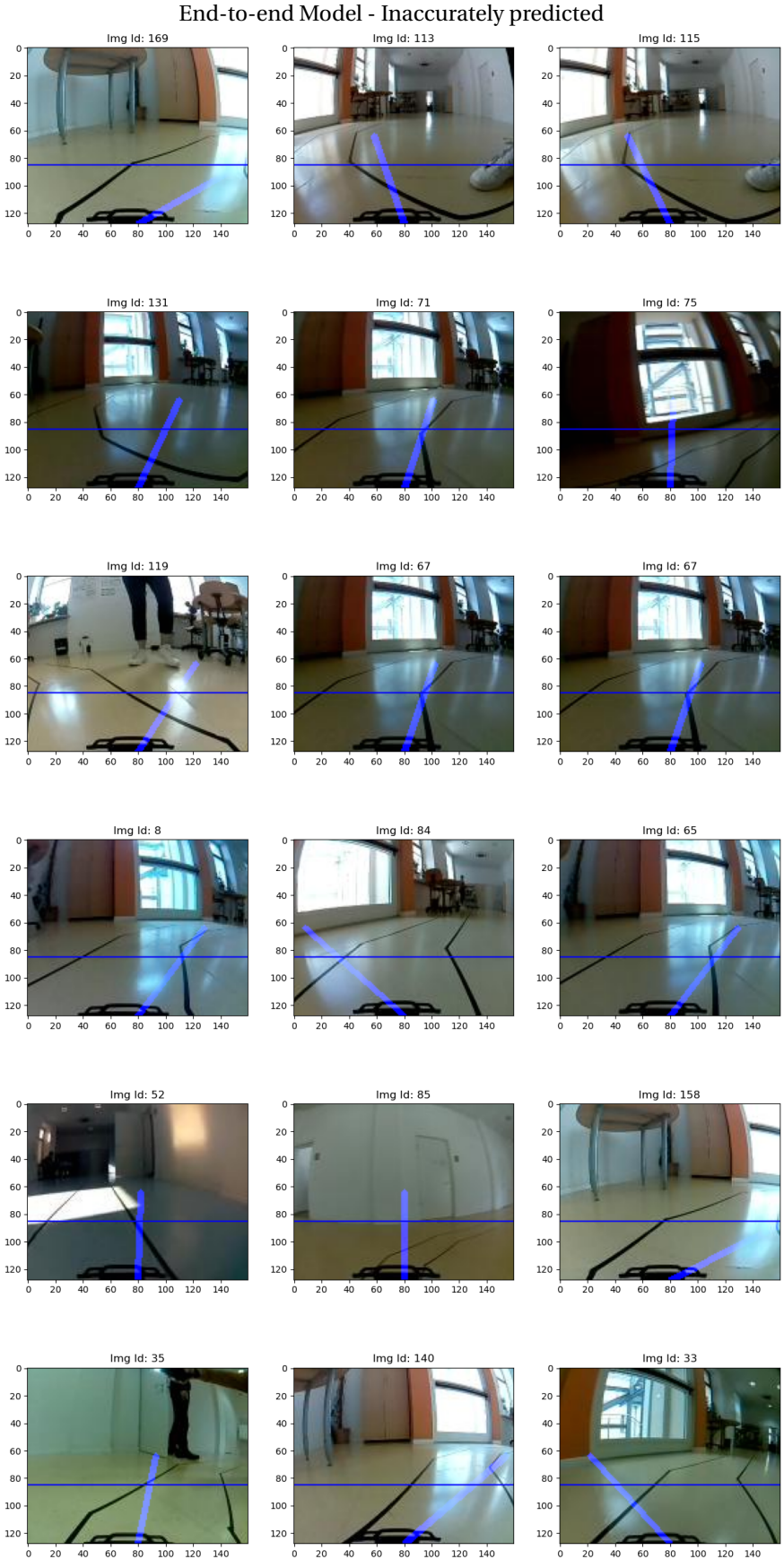


Figure 4.5: Accuracy of steering angle predictions by the different models: a comparison of correct and incorrect predictions

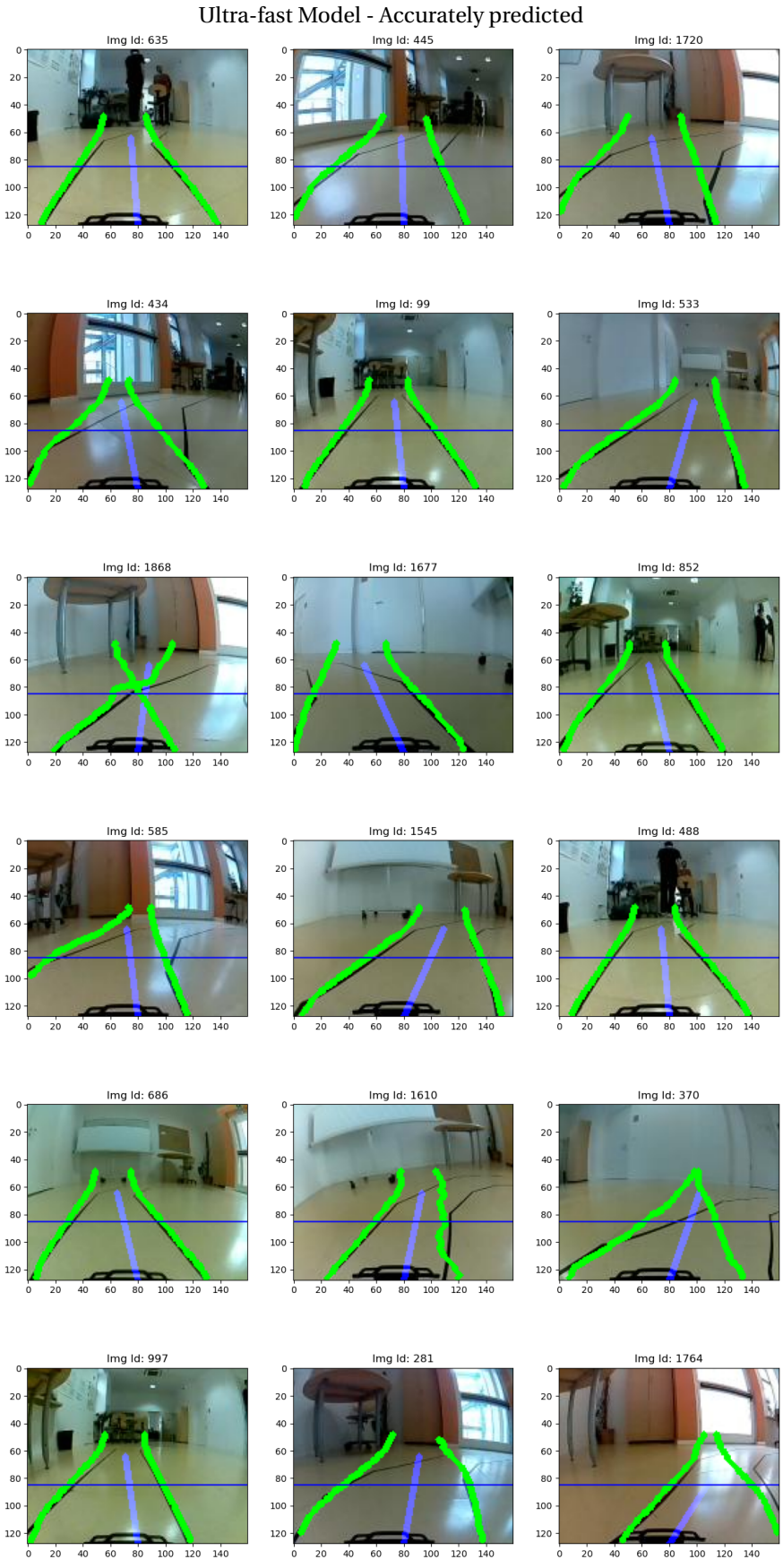


Figure 4.5: Accuracy of steering angle predictions by the different models: a comparison of correct and incorrect predictions

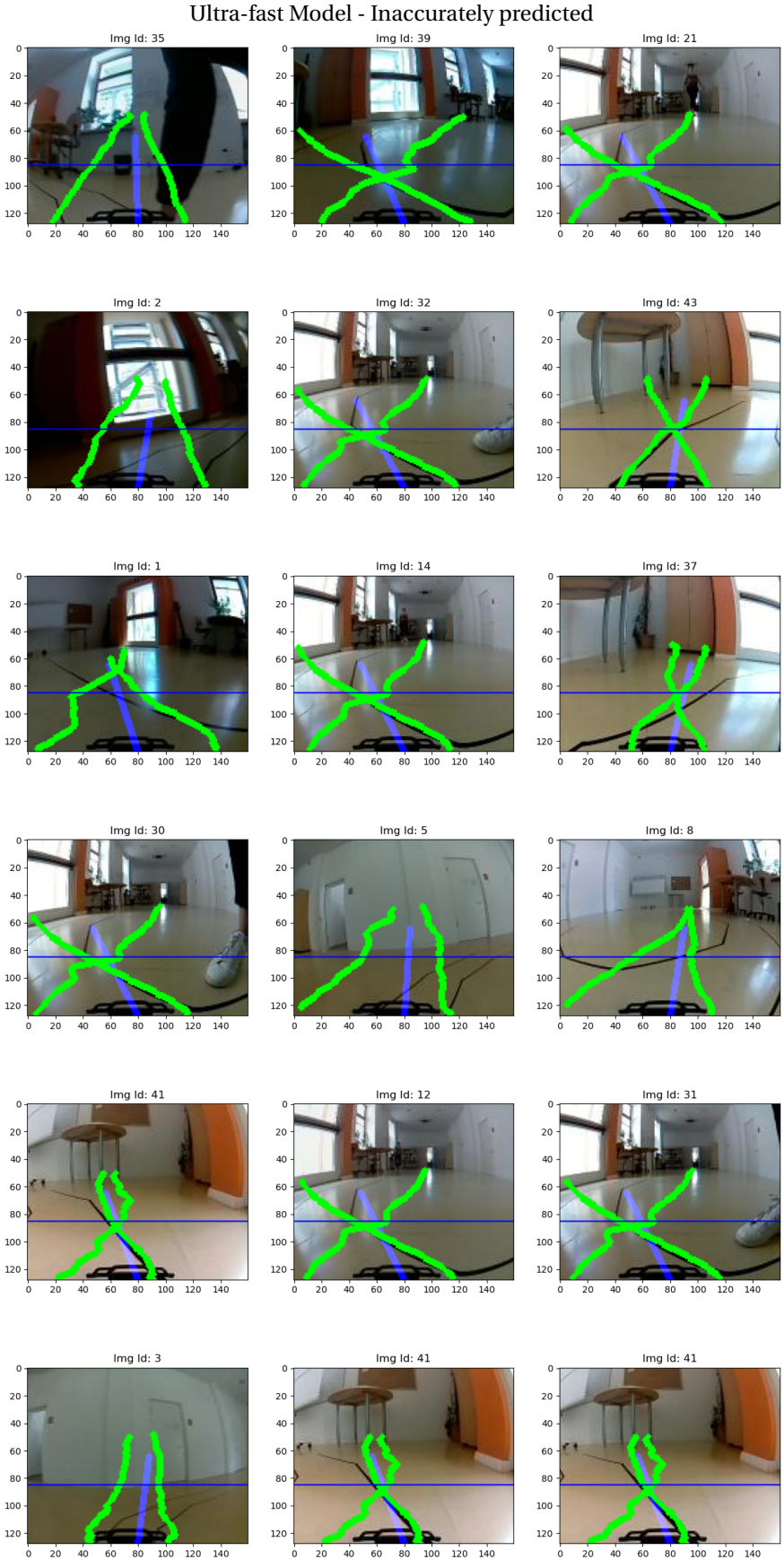


Figure 4.5: Accuracy of steering angle predictions by the different models: a comparison of correct and incorrect predictions

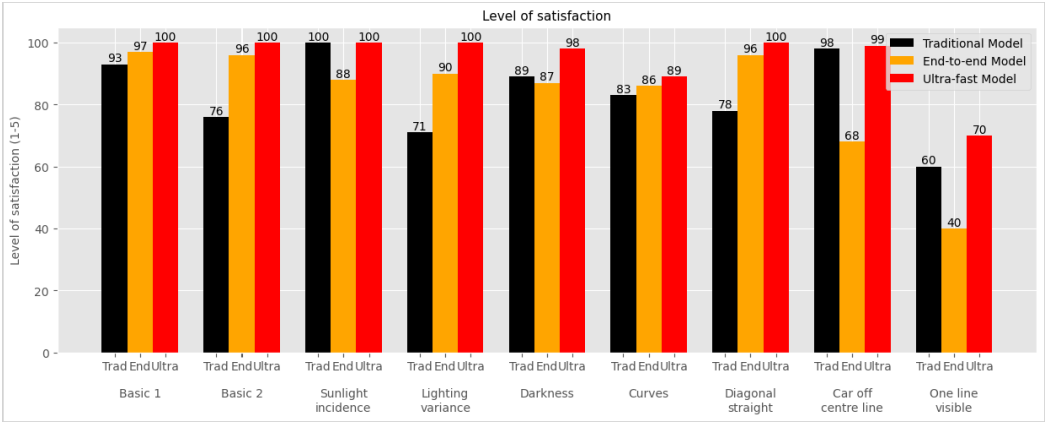


Figure 4.6: Correct angle prediction rates among the different models and scenarios

Based on the results presented in table 4.2 and figure 4.6, the following considerations can be made:

- All three models performed satisfactorily, with the lowest performing model achieving an average accuracy rate of 86% in determining the correct angles.
- In general, the Traditional Model performed worse than the others. During testing, it was noted that this model struggled in certain conditions where there was variation in illumination, causing the car to lose control at the same track marker on each lap.
- The Ultra-fast Model demonstrated superior accuracy across all testing conditions, while the End-to-end Model exhibited inferior performance in certain situations, including instances of bright sunlight, darkness, off-center vehicle position, or absence of visible lines. To enhance the model's performance in these challenging scenarios, additional training and increased data sampling are recommended.
- The End-to-end Model, which directly estimates steering angles, showed higher autonomy than the Traditional Model, despite being trained on data produced by the Traditional Model. Additionally, the End-to-end Model is more sensitive to the distribution of input images and features compared to the Ultra-fast Model.
- The Ultra-fast Model is more complex compared to the others, but even with training labels generated by the first model, it was able to achieve better accuracy in determining angles. The automatic label generation did not compromise the identification of the main direction line.

4.2.2 Memory usage and image processing time

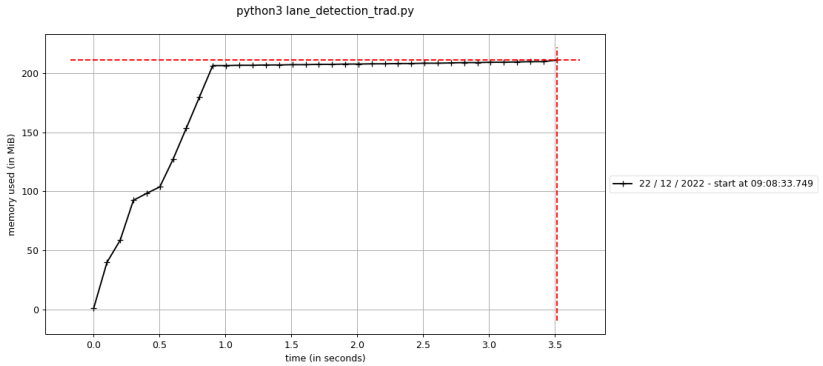
The figure 4.7 and table 4.3 illustrate the estimated image processing time without the use of a GPU and the memory requirements of the models.

The FPS values presented in this study were obtained through computer simulation. To estimate their values, a minimum value of one millisecond per frame was adopted.

It is important to note that the actual FPS may be limited when using a Raspberry Pi camera or any other equipment due to hardware constraints. During our testing, a maximum of 99 frames per second was achieved due to the limitations of the hardware. Therefore, while the simulated FPS values are useful for comparison purposes, the actual performance of the camera may be different in real-world scenarios.

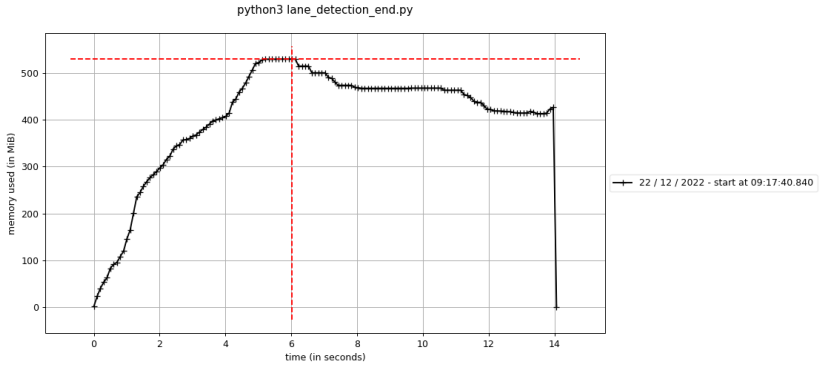
Memory (Mib) x Time (seconds)

Model 1 - Traditional Model



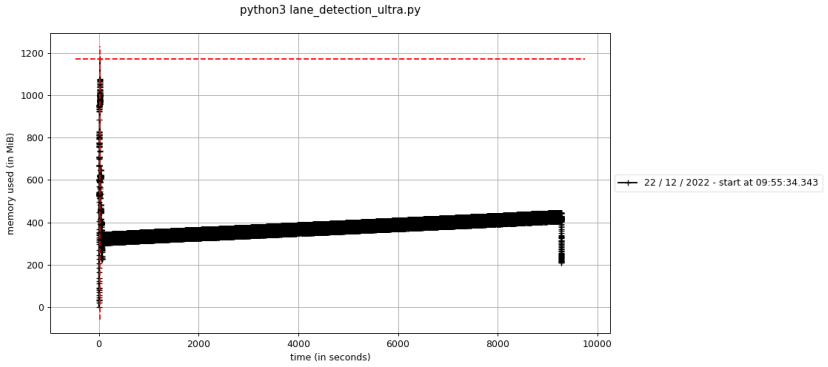
(a)

Model 2 - End-to-end Model



(b)

Model 3 - Ultra=fast Model



(c)

Figure 4.7: Comparison of memory usage and image processing time among models

Models	Model 1 - Traditional	Model 2 – End-to-end	Model 3 - Ultra
Memory (peak)	211 MIB	530 MIB	1170 MIB
Duration	3.5 seconds	14.05 seconds	2h35m
Frames per second	758 FPS	232 FPS	0.18 FPS

Table 4.3: Comparison of memory usage and image processing time among models

The ideal frame rate for lane detection can vary depending on several factors such as the speed of the vehicle, the type of road, the level of detail required in the lane detection, and the computational resources available.

However, a frame rate of 24 to 30 FPS is generally considered sufficient for most lane detection applications. This is supported by the fact that movies are typically encoded at 24 FPS, while TV shows are usually encoded at 25 to 30 FPS. For instance, in [19], a frame rate of 24 FPS is recommended as the configuration parameter for: lane departure systems; traffic sign and light recognition; and vehicle, pedestrian, and cycle detection.

The results showed that the Traditional Model had the best performance in terms of memory usage and processing time. The End-to-end Model also performed well in these criteria, however, the Ultra-fast Model had a less favorable outcome regarding processing time, obtaining a rate of 0.18 frames per second.

Efforts have been undertaken to enhance the number of frames per second:

- Transforming the labels into a grid shape of (20, 100, 2), rather than preserving the original size of the input image. In the previous model, a grid of format (56, 100, 4) was generated, but the labels were saved as images of the same size as the input (128, 160, 3).
- Reducing the model image size from (288, 800, 3) to (96, 256, 3) and making the necessary adjustments to the model). Reducing the size of the data can potentially improve processing time, although information can be lost. For this specific task, this approach was found to be effective, with no negative impact on the results.
- Lighter backbones were tested to improve the efficiency and reduce the computational requirements of deep learning models.
- The model was tested on the PyTorch and TensorFlow libraries. The use of TensorFlow Lite was evaluated across three scenarios (TensorFlow, TensorFlow with Default Quantization, TensorFlow with Full Quantization), however, no improvement in image processing time was observed, only on the amount of memory.

The main findings from these analyses are presented in the table 4.4, which summarizes the modifications made and their impact on processing time and accuracy. It is important to note that accuracy was calculated based on the results of the Traditional Model.

The code and pipeline were optimized, and the model was converted to use the TensorFlow library. Following a promising approach suggested by [20], a substantial reduction in processing time and memory was achieved while maintaining high accuracy by converting the labels to a grid format of (20, 100, 2). This approach yielded results that closely approached the original performance, surpassing the other two models (Traditional and End-to-end) by a significant margin without sacrificing quality.

Additionally, reducing the size of the model image can contribute to an increase in FPS and a slight improvement in accuracy.

ID	Scenario	Description of scenario	Acc	FPS	Memory
1	Ultra-fast	Original code in Pytorch, without code optimisation	85.67%	0.18 FPS	1170 MIB
2	Ultra-fast + grid optimized	Transforming the labels into a grid shape of (20, 100, 2), code and pipeline improvement and adoption of Tensorflow library	84.29%	6.59 FPS	1319 MIB
3	Ultra-fast + grid optimized + reduced image	Changing the size of the model from a shape of (288, 800, 3) to (96, 256, 3)	85.50%	12.98 FPS	1199 MIB
4	Ultra-fast + grid optimized + TfLite	Convert the model to use in Tensorflow Lite (tf-lite runtime), using technique for full quantization (uint8)	32.35%	0.06 FPS	565 MIB
5	Ultra-fast + grid optimized + MobileNetV3	Changing the Resnet-18 backbone to a MobileNetV3-Large backbone	44.77%	11.2 FPS	1279 MIB

Table 4.4: Impact of actions on image processing time, memory and accuracy

The other two attempts(using TfLite or MobileNetV3 as backbone) significantly decreased the accuracy, reaching values below 50%. Although using a lighter backbone increased FPS considerably, they had a notable impact on accuracy. Furthermore, while using Tensorflow Lite significantly reduced memory usage, it led to a significant reduction in both FPS and accuracy.

To develop the lighter models, adjustments were made to the loss function to simplify and adapt them to the new format, following the suggestion of citeKasten2020tflite. The impact on the accuracy of these adjustments was small, as demonstrated by comparing the results of the modified models (Ultra-fast + grid) with the original model (Ultra-fast).

The results for each model generated by the different approaches are presented in Figure 4.8, which includes information about the resources utilized by each model.

4.2.3 Level of satisfaction with scalability of the models

The scalability of the models was evaluated based on the level of satisfaction, using a five-point scale, with 5 indicating the highest level of satisfaction and 1 indicating the lowest. The Ultra-fast Model demonstrated exceptional scalability compared to the other models, achieving a score of 5. On the other hand, the Traditional Model received a lower rating of 2 due to its need to adapt its combination of techniques and configurations to handle various scenarios.

Level of satisfaction:

- Traditional Model: 2
- End-to-end Model: 3
- Ultra-fast Model: 5

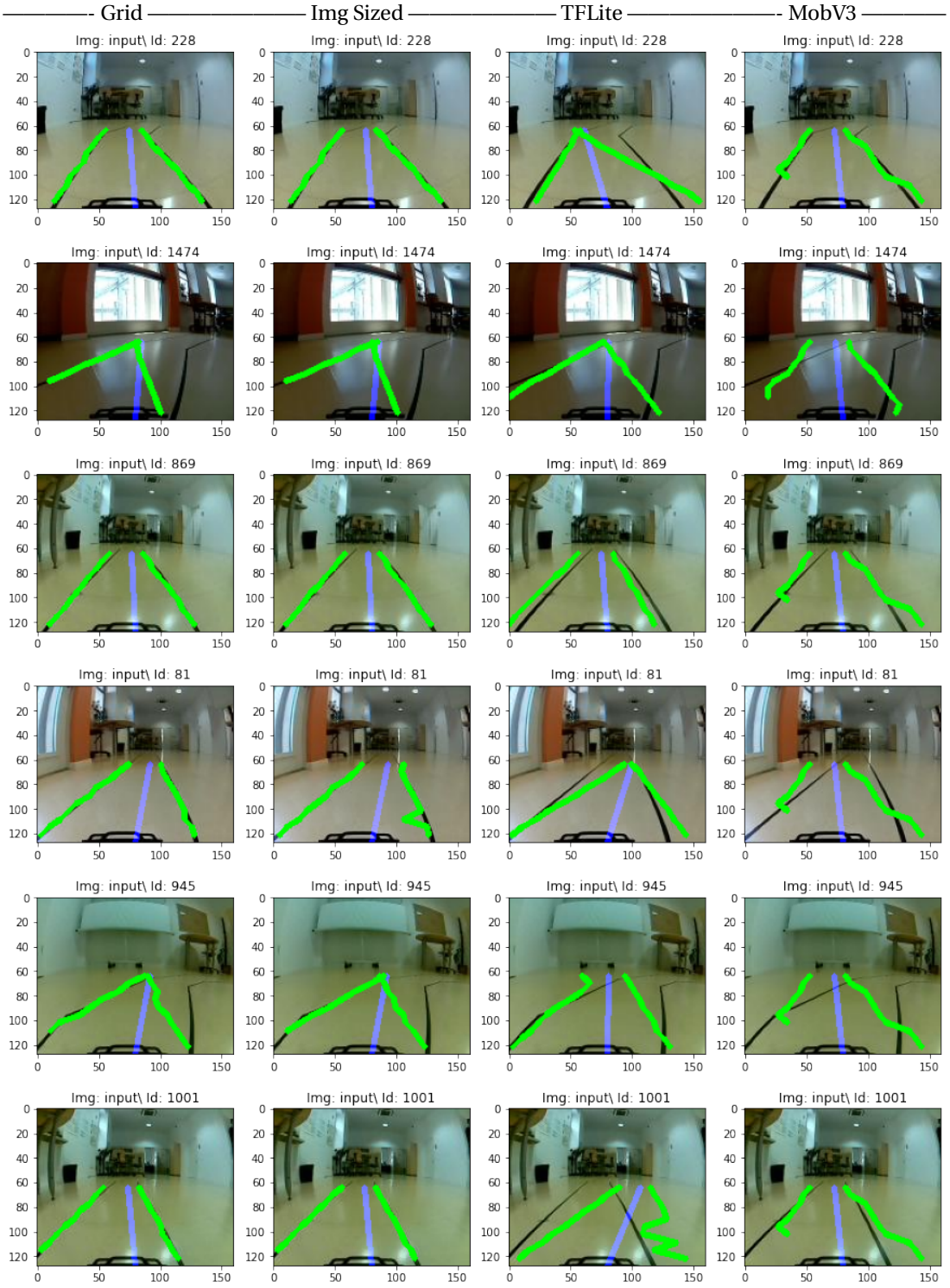


Figure 4.8: Prediction examples of additional Ultra-fast Model variants

4.2.4 Level of satisfaction with the ease of implementation of the model in code

The easy of implementing the models in code was evaluated based on the level of satisfaction, using a five-point scale, with 5 indicating the highest level of satisfaction and 1 indicating the lowest. The End-to-end Model was the easiest to develop and code, receiving a score of 5. In contrast, the traditional and Ultra-fast Models received both lower ratings of 3, as their implementation requires a lot of testing, configurations, and parameters.

Level of satisfaction:

- Traditional Model: 3
- End-to-end Model: 5
- Ultra-fast Model: 3

4.2.5 Level of satisfaction with the required effort to make the model memory-efficient and fast with no GPU

The effort required to optimize the models for memory efficiency and speed was also evaluated based on the level of satisfaction, using a five-point scale, with 5 indicating the highest level of satisfaction and 1 indicating the lowest. The traditional and End-to-end Models delivered exceptional results, achieving a score of 5, with an FPS of over 230. However, the Ultra-fast Model required considerable attention to maintain its performance in scenarios without a GPU, receiving a score of 2.

Level of satisfaction:

- Traditional Model: 5
- End-to-end Model: 5
- Ultra-fast Model: 2

4.2.6 Summary of the results

Figure 4.9 provides a comprehensive overview of the performance of the three models across various metrics, including accuracy, FPS, memory usage, and GPU requirements. Additionally, to further augment the analysis, we have included the results for additional Ultra-fast Models in figure 4.10. These figures collectively offer a clear and concise summary of the models' performance and assist in determining the most suitable model for a particular application.

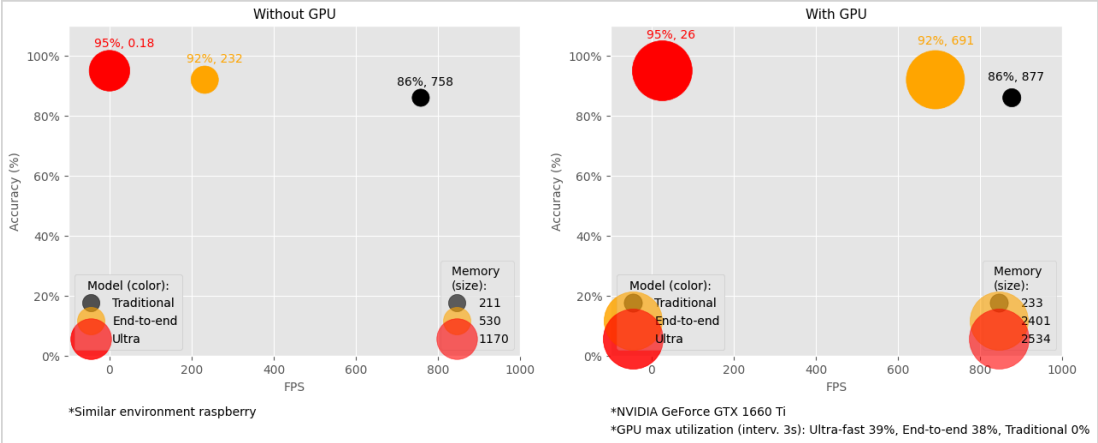


Figure 4.9: Comparison among the performance of the three models

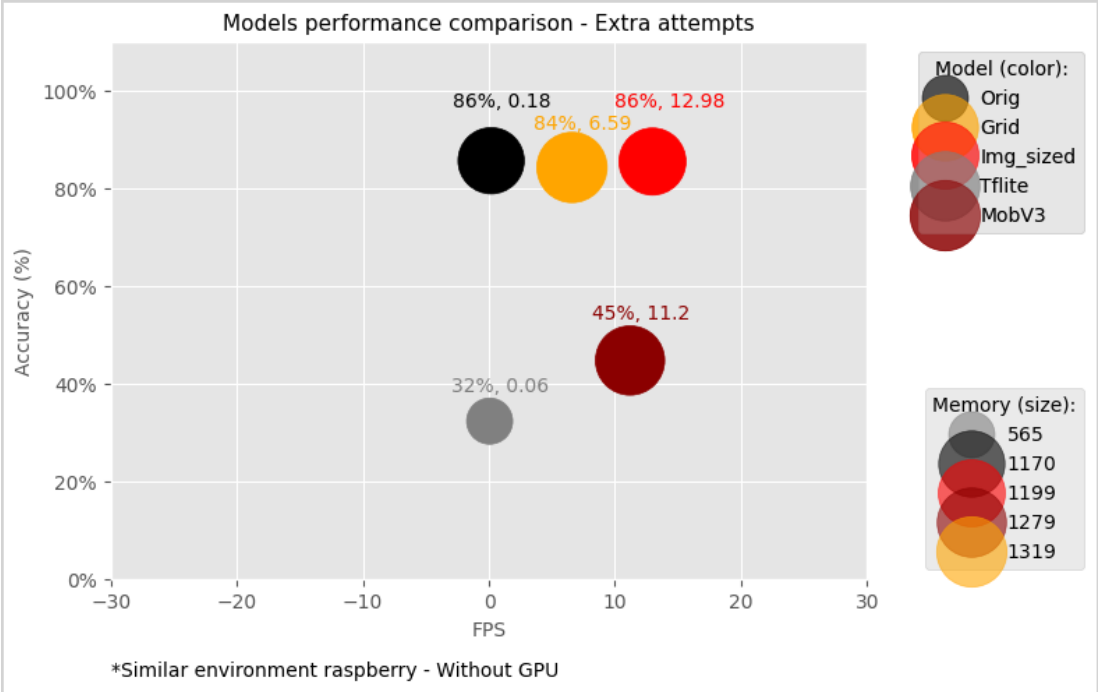


Figure 4.10: Performance comparison of Ultra-fast Models, including additional variants

Complementing the results, the satisfaction levels, which were described in detail in this section are presented in Figure 4.11."

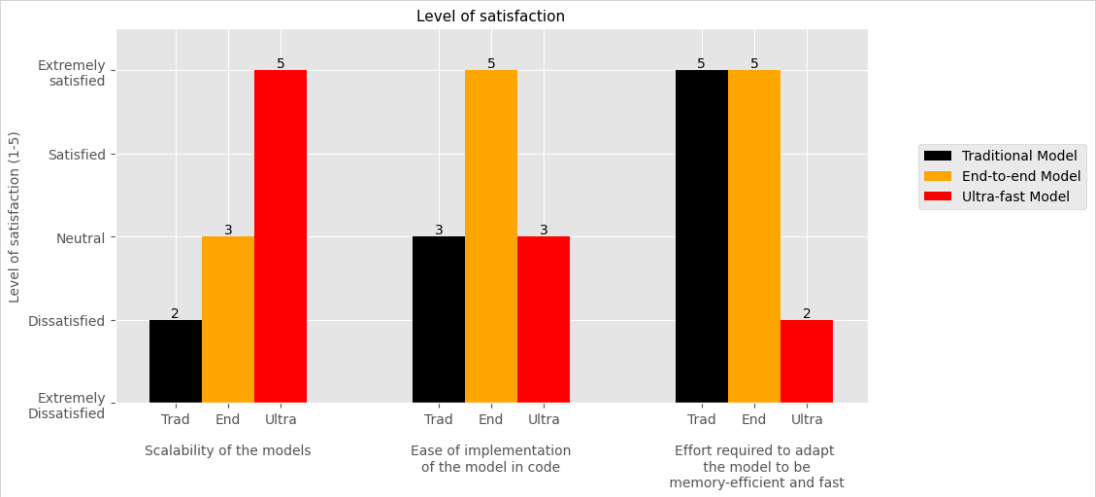


Figure 4.11: Comparison of model satisfaction levels across scalability, ease of implementation, and memory and speed optimization for the three models

4.3 Related work

4.3.1 Related work summary

During the literature mapping process, a total of 52 articles were identified that employed traditional methods for lane detection in smaller devices, reporting a wide range of accuracy (ranging from 63% to 100%) and FPS values (from 6 to 3050 FPS).

In contrast, only 15 studies among the reviewed literature utilized machine learning (ML) methods, indicating a scarcity of research in this area and a lack of solutions for limited memory and low-cost devices. Among these, eight studies either described experiments or used only visual validation, leaving only 7 studies for further analysis.

Out of the seven studies, two ([21] and [22]) predicted steering directions with high accuracy between 86% and 95%. An additional three studies calculated angles using Nvidia and Raspberry Pi devices, each utilizing at least 8GB of GPU for additional power. Of these, two reported their resulting frames per second (FPS) rates: [23] at 22 FPS and [24] at 5-7 FPS, with the latter also reporting an error rate. Only [25] reported accuracy, which was 63%.

The remaining two studies predicted lane boundaries. [26] developed a customized full neural network and tested a miniature car using a laptop. The researchers calculated the F1 score and achieved a value of 98.79%. Although the model predicted lane boundaries, its evaluation was based on the correct prediction of directions. Finally, [27] developed a model based on the classification idea of an ultra-fast lane detection model. They proposed a simplified lane structure loss function and used a lighter backbone (Squeezenet) on a Raspberry 4B CPU or NVIDIA GTX 2070 GPU, achieving an accuracy of 94.46%.

None of the ML models in the literature were compared in terms of their memory consumption and CPU utilization.

4.3.2 Comparisons with related work

Comparing the results from this research with those of the literature is challenging due to the different methods employed, variation in accuracy calculation criteria, and a variety of hardware used.

This reinforces the importance of this study, which provides a thorough comparison of the three different models, isolating the hardware effects, which serves as a valuable reference for selecting a solution for a specific device.

The differences in the methods and criteria used for calculating accuracy in the literature further complicate the comparison, highlighting the need for standardized methods and evaluation metrics in the field. Although standards for lane detection exist on specialized websites, they are not widely used and may not be suitable for real-time experiments using miniature cars and low-cost devices.

Our analysis of the literature highlights the need for new methods and solutions, as demonstrated by Yang2021, who achieved better accuracy and FPS by using a lighter version of the backbone. This finding suggests that future studies should focus on developing more efficient and lightweight models to address the challenges of limited memory and low-cost devices.

4.4 Limitations and Threats to Validity

4.4.1 Limitations of the research

The use of automated approximate labels despite being estimated and not perfect, was found to be useful and effective. The incorporation of these labels into new models resulted in improved performance. However, labels covering the exact line of the track can generate even better results and allowed a complete performance analysis, with precise indicators for accuracy, precision, recall, f2-score, ROC, etc.

4.4.2 Internal validity

In this research, a well-structured methodology was applied to minimize bias and enhance the validity.

Firstly, a systematic mapping study was conducted to gain a comprehensive understanding of previous work on lane detection for miniaturized cars and low-cost devices, providing an accurate picture of the methodologies used.

During the experiment, multiple sets of pictures captured at different times were utilized for both training and testing purposes, and their results were analysed. To avoid mono-method bias, different metrics were used to analyze the results. The results revealed consistency across all three models for all sets and subsets and consistent for the different metrics. The same images were used to test all models.

To avoid overfitting, similar figures were removed from the dataset, and to address imbalanced data, additional images featuring varying track geometry and lighting conditions were incorporated. The study further improved the learning and evaluation of the deep learning models by randomly assigning these images to the training and test sets through stratification.

Additionally, clear evaluation criteria were established for the purpose of evaluating the performance and comparison of the models, avoiding experimenter expectations from interfering in the results.

However, it is important to note that for each of the algorithms a set of parameters and assumptions were adopted, as for example, on the conversion of the steering guides into angles. Different assumptions could have led to variations in the results obtained.

4.4.3 External Validity

This study focuses on miniature cars and low-cost devices, but its implications can be extended to real cars and real-world lane detection.

However, it is crucial to recognize that various factors impact the performance of the models, including the type, quality and size of the dataset, the selection of hyperparameters, and the specific architecture of the model. Therefore, it is important to conduct multiple experiments and compare the results to get a better understanding of the relative performance of the models.

Chapter 5

Final Considerations

This chapter provides a summary of the key results achieved by this thesis, including highlights from the comparison of the three algorithms analyzed herein. In addition, we advance some possible directions for future research.

5.1 Results and Conclusions

The results of this project are three-fold.

Firstly, a systematic literature review was conducted to provide an overview of research in the field, identify trends and gaps, understand the solutions adopted, and assess their challenges and limitations. The results of this study revealed the need for robust comparisons among models that focus on low-cost devices and isolate hardware effects. Few publications have analyzed machine learning models for these devices, and even fewer alternative models have been investigated, particularly those predicting lane boundaries. This study made clear that it is crucial to investigate solutions that reduce decision-making time and delay while producing high-quality results with limited computational resources.

Secondly, three models for lane detection in miniature cars and low-cost devices were implemented: the Traditional Model, based on classical image processing techniques; the End-to-end Model, a CNN that directly predicts angles; and the Ultra-fast Model, which uses a CNN with a ResNet-18 backbone to predict lane boundaries. Each model was analyzed and optimized to enhance its robustness and accuracy in providing results.

Finally, an experiment was designed and executed to compare the performance of these three algorithms. The experiment was conducted in the following manner:

- Data was collected using a miniature car on a designer track in a controlled environment, including additional images with diverse lighting conditions and variations in lane geometry.
- The results of the Traditional Model were used as labels for train the DL models.
- A similar environment was simulated on a laptop and the models were compared in terms of accuracy, memory, GPU, and FPS.
- Additional variations to the Ultra-fast Model were explored and their impact on performance was assessed.

- Subjective information was also produced for the models including: the researcher's satisfaction with the scalability of the models, how easy was to implement the code, and required effort to make the models memory-efficient.

The results of this study demonstrate the suitability of deep learning models for lane recognition tasks, with the two evaluated deep learning models achieving the highest levels of accuracy compared to the traditional method.

Furthermore, the findings confirm our initial hypothesis that the Ultra-fast Model outperforms the other models in terms of accuracy. This is due to the algorithm's optimized structure that enables more precise detection of lanes. However, this model requires more GPU for appropriate processing time and higher FPS rates.

The End-to-end Model is relatively easy to implement and has a comparatively fast processing speed. However, it can be highly influenced by quality of the data used during training. For example, the distribution of the images and their features during training can strongly influence the accuracy of the algorithm. Therefore, it is crucial to ensure the training data used is representative of all types of relevant scenarios and actions.

Implementing these Deep Learning models requires label generation, which can require extra time and effort if the labels are not readily available in a comparable setting. For instance, in this experiment, we had to employ the traditional model to generate the required labels.

In contrast, the Traditional Model achieves high accuracy without requiring training or labels, and it requires less memory than the other two models analysed. However, on the negative side, it is less scalable than the alternatives considered here.

Choosing the best lane recognition model for a specific application depends on several factors, including the available time and resources. Our results provide a useful guide for selecting the appropriate model based on the requirements of memory, GPU, accuracy, and processing time. In addition to these factors, it is important to consider other non-functional requirements such as the time required for code development and label generation.

For a more balanced trade-off between accuracy and memory requirements, the End-to-end Model may be a suitable choice. In addition, it can easily be implemented; if resources are limited and achieving a higher FPS rate is important, the Traditional Model may be a better choice; finally, if achieving high accuracy is the primary objective and there are sufficient resources and time, the Ultra-fast Model is the best choice.

A video demonstrating the results of the three models is available on our GitHub repository at https://gitlab.inf.unibz.it/Rachel.FantiCoelhoLima/intelligent-lane-detection-in-low-cost-devices, where readers can access the relevant files, including code for the implemented algorithms, as well as the results of the experiments conducted here.

5.2 Further Studies

In the sequel, the potential directions for future improvement of each of the algorithms considered in this work are elaborated:

- enhancing the **Traditional Model**'s ability to detect lanes under varying lighting conditions;
- incorporating techniques to handle imbalanced data in the **End-to-end Model**;

- enhancing the **Ultra-fast Model** algorithm to reduce its memory consumption, and to increase processing speed.

Furthermore, one could investigate whether alternative deep learning models could provide for more efficient and directly implementable methods.

Finally, other potential directions of improvement include: developing specialized models specifically for edge detection; exploring new and advanced equipment with improved processing power and lower costs, as well as new backbones and platforms for low-cost devices may also lead to better results.

Bibliography

[1] Daniel Holland-Letz, Matthias Kässer, Benedikt Kloss, and Thibaut Müller. Mobility's future: An investment reality check. https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/mobilitys-future-an-investment-reality-check, 2021. Accessed: 2023-01-16.

[2] Mordor Intelligence. Autonomous (driverless) car market analysis - industry report - trends, size & share. https://www.mordorintelligence.com/industry-reports/autonomous-driverless-cars-market-potential-estimation, 2021. Accessed: 2023-01-16.

[3] Nikolaus Lang, Andreas Herrmann, Markus Hagenmaier, and Maximilian Alexander Richter. Can self-driving cars stop the urban mobility meltdown? 2020.

[4] National Safety Council. Advanced driver assistance systems. https://injuryfacts.nsc.org/motor-vehicle/occupant-protection/advanced-driver-assistance-systems/data-details/, 2020. Accessed: 2023-01-16.

[5] Jozsef Suto. Real-time lane line tracking algorithm to mini vehicles. *Transport and Telecommunication*, 22(4):461–470, 2021.

[6] Gurjashan Singh Pannu, Mohammad Dawud Ansari, and Pritha Gupta. Design and implementation of autonomous car using raspberry pi. *International Journal of Computer Applications*, 113(9), 2015.

[7] Jigang Tang, Songbin Li, and Peng Liu. A review of lane detection methods based on deep learning. *Pattern Recognition*, 111:107623, 2021.

[8] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 286–291. IEEE, 2018.

[9] Dong Wu, Man-Wen Liao, Wei-Tian Zhang, Xing-Gang Wang, Xiang Bai, Wen-Qing Cheng, and Wen-Yu Liu. Yolop: You only look once for panoptic driving perception. *Machine Intelligence Research*, pages 1–13, 2022.

[10] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[11] Zequn Qin, Huanyu Wang, and Xi Li. Ultra fast structure-aware deep lane detection. In *European Conference on Computer Vision*, pages 276–291. Springer, 2020.

[12] Qin Zou, Hanwen Jiang, Qiyu Dai, Yuanhao Yue, Long Chen, and Qian Wang. Robust lane detection from continuous driving scenes using deep neural networks. *IEEE transactions on vehicular technology*, 69(1):41–54, 2019.

[13] Mohsen Ghafoorian, Cedric Nugteren, Nóra Baka, Olaf Booij, and Michael Hofmann. El-gan: Embedding loss driven generative adversarial networks for lane detection. In *proceedings of the european conference on computer vision (ECCV) Workshops*, pages 0–0, 2018.

[14] Yan Liu, Jingwen Wang, Yujie Li, Canlin Li, and Weizheng Zhang. Lane-gan: A robust lane detection network for driver assistance system in high speed and complex road conditions. *Micromachines*, 13(5):716, 2022.

[15] Tu Zheng, Yifei Huang, Yang Liu, Wenjian Tang, Zheng Yang, Deng Cai, and Xiaofei He. Clrnet: Cross layer refinement network for lane detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 898–907, 2022.

[16] Lizhe Liu, Xiaohao Chen, Siyu Zhu, and Ping Tan. Condlanenet: a top-to-down lane detection framework based on conditional convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3773–3782, 2021.

[17] Cheng Han, Qichao Zhao, Shuyi Zhang, Yinzi Chen, Zhenlin Zhang, and Jinwei Yuan. Yolopv2: Better, faster, stronger for panoptic driving perception. *arXiv preprint arXiv:2208.11434*, 2022.

[18] David Tian. Deeppicar — part 5: Autonomous lane navigation via deep learning. https://towardsdatascience.com/deeppicar-part-5-lane-following-via-deep-learning-d93acdce6110, 2019. Accessed: 2023-01-16.

[19] Prahanth Viswanath, Kedar Chitnis, Pramod Swami, Mihir Mody, Sujith Shivalingappa, Soyeb Nagori, Manu Mathew, Kumar Desappan, Shyam Jagannathan, Deepak Poddar, et al. A diverse high-performance platform for advanced driver assistance system (adas) applications. *Texas Instruments, Inc*, pages 1–16, 2016.

[20] Markus Kasten. Tensorflow 2 / lite implementation of ultra-fast structure-aware lane detection. https://github.com/markus-k/ultrafast-lane-detection-tf2, 2020. Accessed: 2023-01-16.

[21] Uvais Karni, S. Shreyas Ramachandran, K. Sivaraman, and A. K. Veeraraghavan. Development of autonomous downscaled model car using neural networks and machine learning. In *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1089–1094, 2019.

[22] Anita Chaudhari, Dhvani Shah, Kiran Mungekar, and Vidhan Wani. Design and implementation of car for smart cities—intelligent car prototype. In *Soft Computing and Signal Processing: Proceedings of ICSCSP 2018, Volume 2*, pages 485–494. Springer, 2019.

[23] Khanh Du Nguyen Tu, Hoang Dung Nguyen, and Thanh Hai Tran. Vision based steering angle estimation for autonomous vehicles. In *2020 International Conference on Advanced Technologies for Communications (ATC)*, pages 187–192. IEEE, 2020.

[24] Shenbagaraj Kannapiran and Spring Berman. Go-chart: A miniature remotely accessible self-driving car robot. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2265–2272, 2020.

[25] WeiHang Feng, Yun Pei, Bin Hai, WuLing Huang, XiaoYan Gong, and Fenghua Zhu. Autonomous rc-car for education purpose in istem projects. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1905–1909, 2018.

[26] Md. Al-Masrur Khan, Seong-Hoon Kee, Niloy Sikder, Md. Abdullah Al Mamun, Fatima Tuz Zohora, Md. Tariq Hasan, Anupam Kumar Bairagi, and Abdullah-Al Nahid. A vision-based lane detection approach for autonomous vehicles using a convolutional neural network architecture. In *2021 Joint 10th International Conference on Informatics, Electronics & Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pages 1–10, 2021.

[27] Dongdong Yang, Wenzhi Bao, and Kai Zheng. Lane detection of smart car based on deep learning. In *Journal of Physics: Conference Series*, volume 1873, page 012068. IOP Publishing, 2021.