



**HAL**  
open science

# Towards visual urban scene understanding for autonomous vehicle path tracking using GPS positioning data.

Citlalli Gamez Serna Gamez Serna

## ► To cite this version:

Citlalli Gamez Serna Gamez Serna. Towards visual urban scene understanding for autonomous vehicle path tracking using GPS positioning data.. Other [cs.OH]. Université Bourgogne Franche-Comté, 2019. English. NNT : 2019UBFCA004 . tel-02160966

**HAL Id: tel-02160966**

**<https://theses.hal.science/tel-02160966v1>**

Submitted on 20 Jun 2019

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

**THÈSE DE DOCTORAT DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ**  
**PRÉPARÉE À L'UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD**

École doctorale n°37  
Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Informatique

par

CITLALLI GÁMEZ SERNA

**Towards Visual Urban Scene Understanding for Autonomous Vehicle Path  
Tracking using GPS Positioning Data**

Thèse présentée et soutenue à Belfort, le 29 avril 2019

Composition du Jury :

VASSEUR PASCAL	Professeur à l'Université de Rouen	Rapporteur
LAUFFENBURGER JEAN-PHILIPPE	Professeur à l'Université de Haute Alsace	Rapporteur
TRÉMEAU ALAIN	Professeur à l'Université Jean Monnet	Examineur
KHOUDOUR LOUAHDI	Directeur de Recherche à CEREMA - Laboratoire Expertise Transports de Toulouse	Examineur
ABBAS-TURKI ABDELJALIL	Maître de Conférences à l'Université de Technologie de Belfort-Montbéliard	Examineur
RUICHEK YASSINE	Professeur à l'Université de Technologie de Belfort-Montbéliard	Directeur de thèse



**Titre :** Towards Visual Urban Scene Understanding for Autonomous Vehicle Path Tracking using GPS Positioning Data

**Mots-clés :** Conduite autonome, Perception de l'environnement, CNN, Segmentation sémantique, Segmentation d'instance, Classification d'image, Panneaux de signalisation, Estimation de courbure, Suivi de trajectoire, Contrôle latéral.

**Résumé :**

Cette thèse de doctorat s'intéresse au suivi de trajectoire basé sur la perception visuelle et la localisation en milieu urbain. L'approche proposée comprend deux systèmes. Le premier concerne la perception de l'environnement. Cette tâche est effectuée en utilisant des techniques d'apprentissage profond pour extraire automatiquement les caractéristiques visuelles 2D et utiliser ces derniers pour apprendre à distinguer les différents objets dans les scénarios de conduite. Trois techniques d'apprentissage approfondi sont adoptées : la segmentation sémantique pour assigner chaque pixel d'une image à une classe, la segmentation d'instance pour identifier les instances séparées de la même classe et la classification d'image pour reconnaître davantage les étiquettes spécifiques des instances. Ici, notre système considère 15 classes d'objets et reconnaît les panneaux de signalisation. Le deuxième système fait référence au suivi de chemin numérisé. Dans un premier temps, le véhicule équipé enregistre d'abord l'itinéraire avec un système de vision

stéréo et un récepteur GPS (étape d'apprentissage ou numérisation du chemin). Ensuite, le système proposé analyse hors ligne la trajectoire GPS et identifie exactement les emplacements des courbes dangereuses (brusques) et les limitations de vitesse via les données visuelles. Enfin, une fois que le véhicule est capable de se localiser lui-même durant la phase de suivi de chemin, le module de contrôle du véhicule piloté avec notre algorithme de négociation de vitesse, prend en compte les informations extraites et calcule la vitesse idéale à exécuter. Grâce aux résultats expérimentaux des deux systèmes, nous prouvons que le premier est capable de détecter et de reconnaître précisément les objets d'intérêt dans les scénarios urbains, tandis que le suivi de trajectoire réduit significativement les erreurs latérales entre le trajet appris et le trajet parcouru. Nous soutenons que la fusion des deux systèmes améliorera le suivi de chemin pour prévenir les accidents ou assurer la conduite autonome.

**Title:** Towards Visual Urban Scene Understanding for Autonomous Vehicle Path Tracking using GPS Positioning Data

**Keywords:** Autonomous driving, Environment perception, CNN, Semantic segmentation, Instance segmentation, Image classification, Traffic signs, Curvature estimation, Path tracking, Lateral control.

**Abstract:**

This PhD thesis focuses on developing a path tracking approach based on visual perception and localization in urban environments. The proposed approach comprises two systems. The first one concerns environment perception. This task is carried out using deep learning techniques to automatically extract 2D visual features and use them to learn in order to distinguish the different objects in the driving scenarios. Three deep learning techniques are adopted: semantic segmentation to assign each image pixel to a class, instance segmentation to identify separated instances of the same class and, image classification to further recognize the specific labels of the instances. Here our system segments 15 object classes and performs traffic sign recognition. The second system refers to path tracking. In order to follow a path, the equipped vehicle first travels and records the route

with a stereo vision system and a GPS receiver (learning step). The proposed system analyses off-line the GPS path and identifies exactly the locations of dangerous (sharp) curves and speed limits. Later after the vehicle is able to localize itself, the vehicle control module together with our speed negotiation algorithm, takes into account the information extracted and computes the ideal speed to execute. Through experimental results of both systems, we prove that, the first one is capable to detect and recognize precisely objects of interest in urban scenarios, while the path tracking one reduces significantly the lateral errors between the learned and traveled path. We argue that the fusion of both systems will ameliorate the tracking approach for preventing accidents or implementing autonomous driving.



# ACKNOWLEDGEMENTS

I would like to express my special appreciation and thanks to my advisor Dr. Yassine Ruichek for all his help and guidance during my research. His friendly support, patience, regular suggestions and valuable feedback's were very important to the successful completion of this work.

I would like to acknowledge my thesis committee, Pascal Vasseur, Jean-Philippe Lauffenburger, Alain Trémeau, Khoudour Louahdi, and Abdeljalil Abbas-Turki, for accepting and evaluating my thesis work.

Exceptional thanks to Consejo Nacional de Ciencia y Tecnología (CONACYT) for the financial support given during my studies with its program Gobierno Francés 2015. CONACYT is the Mexican entity in charge of promoting scientific and technological activities offering grants to Mexican students to pursue a degree inside the country or abroad.

A special thanks to all the members of the laboratory for encouraging my research and for allowing me to grow as a research scientist. I would also like to thank Yongliang Qiao, Alexandre Lombard, Tao Yang and Jocelyn Buisson for all the support, suggestions and ideas provided every time I needed them.

Also, I would like to express my gratitude to Martin Geissmann for all his unconditional support, during my studies. I would have been harder to make it through the final stage without his help. At the same time, thanks to my friends Ana Karen Gárate Escamilla and Alexis Godart for making my stay in Europe more pleasant.

Most importantly, I dedicate this work to my family for their support during all these hard years away from them.



# SOMMAIRE

<b>I</b>	<b>Context and Problems</b>	<b>1</b>
<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Background . . . . .	3
1.2	Problem definition . . . . .	6
1.3	Objectives . . . . .	9
1.4	Thesis organization . . . . .	10
<b>II</b>	<b>Contribution</b>	<b>13</b>
<b>2</b>	<b>Proposed datasets</b>	<b>15</b>
2.1	Introduction . . . . .	15
2.2	European Traffic Signs Dataset (ETSD) . . . . .	16
2.2.1	Introduction . . . . .	16
2.2.2	Data definition . . . . .	17
2.2.3	Data collection - UTBM dataset . . . . .	21
2.2.4	Data organization . . . . .	22
2.2.5	Conclusions . . . . .	27
2.3	UTBM-2 - A semantic and instance segmentation dataset . . . . .	28
2.3.1	Introduction . . . . .	28
2.3.2	Data specifications . . . . .	29
2.3.3	Data labeling . . . . .	29
2.3.4	Dataset splitting and formatting . . . . .	31
2.3.5	Conclusions . . . . .	32
2.4	Extended GTSD - Traffic Sign Detection Dataset . . . . .	34
2.4.1	Introduction . . . . .	34



2.4.2	Data definition . . . . .	35
2.4.3	Data labeling . . . . .	36
2.4.4	Conclusions . . . . .	37
2.5	Conclusions . . . . .	38
<b>3</b>	<b>Traffic sign classification</b>	<b>39</b>
3.1	Introduction . . . . .	39
3.2	Related works . . . . .	40
3.3	Convolutional Neural Networks for traffic sign classification . . . . .	42
3.3.1	LeNet-5 . . . . .	42
3.3.2	IDSIA model . . . . .	43
3.3.3	URV model . . . . .	44
3.3.4	CNN with asymmetric kernels . . . . .	44
3.3.5	CNN 8-Layers . . . . .	45
3.4	Proposed CNN (Class_CNN) . . . . .	46
3.5	Experimental results . . . . .	48
3.5.1	Performance comparisons without data augmentation . . . . .	49
3.5.2	Performance comparisons with data augmentation . . . . .	52
3.5.2.1	Proposed Class_CNN . . . . .	53
3.5.2.2	CNNs comparisons . . . . .	55
3.5.3	Discussion . . . . .	55
3.6	Conclusion and future works . . . . .	60
<b>4</b>	<b>Visual perception system for urban environments</b>	<b>63</b>
4.1	Introduction . . . . .	63
4.2	Related works . . . . .	65
4.2.1	Semantic segmentation . . . . .	66
4.2.2	Traffic Sign Recognition . . . . .	68
4.3	Visual perception system . . . . .	70
4.3.1	Environment perception through semantic segmentation approaches	71
4.3.1.1	SegNet . . . . .	72
4.3.1.2	Pyramid Scene Parsing Network (PSPNet) . . . . .	73

4.3.1.3	DeepLab . . . . .	73
4.3.1.4	Bilateral Segmentation Network (BiSeNet) . . . . .	75
4.3.2	Traffic sign recognition . . . . .	76
4.3.2.1	Traffic sign detection . . . . .	77
4.3.2.2	Traffic sign refinement and classification . . . . .	79
4.4	Experimental results . . . . .	81
4.4.1	Datasets . . . . .	81
4.4.1.1	Semantic segmentation dataset . . . . .	81
4.4.1.2	Traffic sign detection and classification datasets . . . . .	82
4.4.2	Training . . . . .	84
4.4.2.1	Semantic segmentation approaches . . . . .	84
4.4.2.2	Traffic sign detection . . . . .	85
4.4.2.3	Traffic sign classification . . . . .	87
4.4.3	Semantic segmentation evaluation . . . . .	88
4.4.4	Traffic sign recognition evaluation . . . . .	91
4.4.4.1	Classification performance . . . . .	92
4.4.4.2	Detection performance . . . . .	93
4.5	Conclusions and future works . . . . .	100
<b>5</b>	<b>Dynamic Speed Adaptation System for Path Tracking Based on Curvature Information and Speed Limits</b> . . . . .	<b>105</b>
5.1	Introduction . . . . .	105
5.2	Related works . . . . .	108
5.2.1	Intelligent Speed Adaptation systems . . . . .	108
5.2.2	Curve speed estimation . . . . .	108
5.3	Steering control algorithms for path tracking . . . . .	110
5.3.1	Pure Pursuit . . . . .	110
5.3.2	Stanley method . . . . .	111
5.3.3	Alice method . . . . .	112
5.3.4	Lombard method . . . . .	112
5.4	Dynamic Speed Adaptation (DSA) . . . . .	113

5.4.1	GPS pre-processing . . . . .	114
5.4.2	Speed limit database . . . . .	115
5.4.3	Curve speed estimation . . . . .	116
5.4.3.1	Sharp curve estimation . . . . .	116
5.4.3.2	Speed computation for sharp curves . . . . .	119
5.4.4	Speed negotiation algorithm . . . . .	121
5.5	Results . . . . .	124
5.6	Conclusion and Future Works . . . . .	130
<b>III</b>	<b>Conclusions and Future Works</b>	<b>133</b>
<b>6</b>	<b>Conclusions and Future works</b>	<b>135</b>
6.1	Conclusions . . . . .	135
6.2	Future works . . . . .	137
<b>IV</b>	<b>Annexes</b>	<b>163</b>
<b>A</b>	<b>Publications</b>	<b>165</b>
A.1	Conferences . . . . .	165
A.2	Journals . . . . .	165



# CONTEXT AND PROBLEMS



# INTRODUCTION

## 1.1/ BACKGROUND

Since the last century, automated driving has been one of the research topics that industry and academia have focused on due to the advantages it can provide. Among them are :

- Safety. Reduce the number of accidents cause by human errors.
- Efficiency. Increase the transport system efficiency while reducing traffic jams and transportation time.
- Environmental impacts. Decrease energy consumption and vehicle emissions due to the transportation efficiency.
- Comfort. Allows the user to do something else than driving.
- Social inclusion. Enables mobility for everyone, including the elderly and handicapped people.

The first idea dates back to the 1930s when the General Motors (GM) Corporation had an exhibition about self driving cars. Later, during the 1950s, GM implemented an automatic system for speed control and variable spacing [2]. The cars with such systems were able to navigate on highways with the use of receivers able to detect special circuits installed in the road. Nevertheless, due to intrusive infrastructure modifications and to the limited drivable area, other parties were motivated and joined the research in the field. With the advances in imaging sensors and computer capabilities, in Japan 1977, the MITI's Mechanical Engineering Laboratory was able to control automatically the vehicle's steering wheel with the use of binocular machine vision at 30 km/h [5]. However, it was until 1987, when Ernst Dickmanns achieved an important milestone with the VaMoRs vehicle [14] (see Fig. 1.1), a Mercedes-Benz van modified to control the steering wheel, throttle, and brakes through computer commands based on image sequences. VaMoRs van demonstrated to be the first car being able to drive by itself with computer vision



FIGURE 1.1 – VaMoRs vehicle and a view of its interior.<sup>1</sup>

guidance at speeds up to 96 km/h in highway. Since then, computer vision has been adopted to perceive the environment for autonomous vehicles.

Subsequently to Dickmanns breakthrough, numerous efforts by academia and automotive industry have been made around the world to construct a vehicle with high level of autonomy. The Defense Advanced Research Projects Agency (DARPA) Challenges [30, 26, 46] were the most important events that push many research teams to build driverless vehicles facing real scenarios. For the first DARPA grand challenge, launched in 2003 to navigate 142 miles through the Mojave desert, 107 teams were registered and 15 raced, but none of the cars navigated more than 5% of the course. Because of that, the grand challenge was repeated in 2005 with 195 teams registered, racing 23, and this time having a winner, the Stanford's robot car "Stanley". Due to this success, the DARPA urban challenge [46] was organized in 2006 to drive 97 km in a urban environment interacting with other moving vehicles and obeying the California Driving Handbook. In this challenge, 89 teams were registered from which only 11 were selected to compete. The winner of the urban challenge was a vehicle named "Boss" developed by the Tartan Racing Team composed of students, staff and researchers from several organizations including Carnegie Mellon University, General Motors, Caterpillar, Continental and Intel.

Motivated by the DARPA challenge results, many research groups continued improving their vehicles, leading to progressions in the field and emerging companies. Up to date, some of the projects already implemented, the ones in progress and future estimations can be seen in Fig. 1.2.

Nonetheless, in order to standardize a common definition of what is considered autonomous, in 2014, the Society of Automotive Engineers (SAE) defined 5 levels of vehicle automation to bring better clarity and use them as a standard classification tool [97]. This definition was based on the earlier work made by the US Department of Transportation's

1. <https://medium.com/twentybn/germany-asleep-at-the-wheel-d800445d6da2>

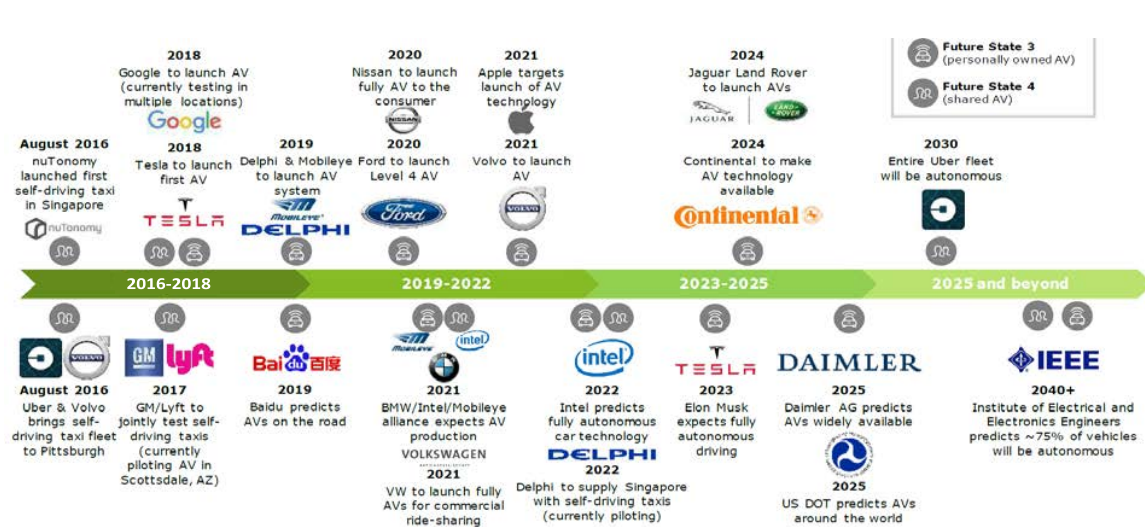


FIGURE 1.2 – Autonomous vehicles : adoptions and applications extracted from the report study [176].

National Highway Traffic Safety Administration (NHTSA) in 2013, which defined 4 levels having an extensive definition for the last one. Regarding the European Union, the German Federal Highway Research Institute (BAST) issued as well 5 levels [70]. A summary of the 3 notations is expressed in Table 1.1. This thesis will refer to the SAE's notation when talking about the different levels.

To this end, we know that some Advanced Driver Assistance Systems (ADAS) already implemented in commercial vehicles like Adaptive Cruise Control, Lane-Keeping Assist and Automatic Emergency Braking, correspond to level 2 since the system is able to control the steering or speed based on sensor measurements like distance or visual lane recognition. Moreover level 3 vehicles are able to perceive the environment and operate under certain conditions, but require the driver to stay alerted in case of an unexpected event that will require to take over the control. Currently, in our knowledge, some commercial vehicles have reached level 3 (Tesla, Audi), but the only one promoted in the market with this level, is the Audi A8 with its AI Traffic Jam Pilot<sup>2</sup>. Level 4 vehicles already exist (taxis, low-speed shuttles) but are pilot projects restricted to operate under certain conditions (areas, weather, speeds) and analyzed with continuous test evaluations to prove their efficiency. SAE level 5 vehicles will not have operational restrictions and will be able to navigate anywhere at any time.

2. [https://www.youtube.com/watch?v=WsiUwq\\_M8IE&feature=youtu.be](https://www.youtube.com/watch?v=WsiUwq_M8IE&feature=youtu.be)



TABLE 1.1 – BAsT, NHTSA and SAE levels of driving automation. System refers to the driver assistance system, combination to driver assistance systems, or automated driving systems, as appropriate.

BAsT level	NHTSA level	SAE level	SAE Name	SAE Narrative definition	Execution of steering and acceleration/deceleration	Monitoring of driving environment	Fallback performance of dynamic driving task	System capability (driving modes)
Driver only	0	0	No Automation	the fulltime performance by the human driver of all aspects of the dynamic driving task, even when enhanced by warning or intervention systems	Human	Human	Human	n/a
Assisted	1	1	Driver Assistance	the driving mode-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the human driver perform all remaining aspects of the dynamic driving task	Human and System	Human	Human	Some driving modes
Partially automated	2	2	Partial Automation	the driving mode-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment and with the expectation that the human driver perform all remaining aspects of the dynamic driving task	System	Human	Human	Some driving modes
Highly automated	3	3	Conditional Automation	the driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task with the expectation that the human driver will respond appropriately fo a request to intervene	System	System	Human	Some driving modes
Fully automated	3/4	4	High Automation	the driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task, even if a human driver does not respond appropriately to a request to intervene	System	System	System	Some driving modes
	3/4	5	Full Automation	the full-time performance by an automated driving system of all aspects of the dynamic driving task under all roadway and environmental conditions that can be managed by a human driver	System	System	System	All driving modes

## 1.2/ PROBLEM DEFINITION

In order to understand what problems the research community is facing to achieve level 5 (full automation), we need to briefly introduce how automated vehicles work, the status of these technologies until now and the future plans to make the dream come true.

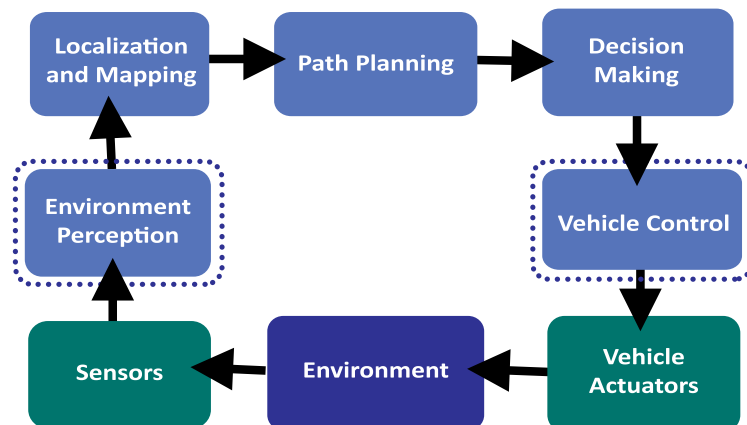


FIGURE 1.3 – Overview of the autonomous navigation process. Light blue blocks indicate the five main components for autonomy. The light blue blocks with additional dotted boxes make reference to the components this thesis focus on for contributions.

The system architecture of autonomous vehicle navigation comprises five main components (light blue blocks in Fig. 1.3) : Environment perception, Localization and Mapping, Path Planning, Decision Making and Vehicle Control [185]. Environment perception requires to interpret the vehicle’s surroundings with the use of several sensors measurements ; in other words, it needs to detect and quantify all relevant elements in the environment similar to how humans do. Localization and Mapping has the function to locate the vehicle globally and locally regarding a reference point and to map the perceived environment with the sensor information. Once the vehicle knows what is around and where it is, the Path Planning module requires to calculate the possible routes to navigate safely. From all the proposed outputs from the Path Planning module, the Decision Making module needs to choose the optimal one taking into account the perceived information and respecting driving rules. Then, the vehicle control module will compute the appropriate acceleration and steering wheel angle commands to pass them to the vehicle actuators to follow the chosen route.

The five components just described, make reference to an automated vehicle able to perceive the environment, make decisions and execute the appropriate commands in a continuously manner all by itself. The final aim in this field, is to reach level 5 (full automation) providing safety not only for the passenger, but for every agent involved in the driving scenario. The current state of the art have reached level 3 for commercial vehicles and level 4 for some companies pilot projects (refer to [185] for a more detailed and comprehensive overview).

In the efforts to demonstrate the advantages of automated vehicles, even with a driver behind the steering wheel, several accidents (comprising the leading companies Google, Tesla, Uber, Navya) have initiated concerns regarding the technology's limitations, reliability, appropriate driving scenarios, and the drivers' understanding and capability of using the technology safely [164]. Most of the incidents, either involved error on the part of the other driver, or failures in perceiving the environment and/or driver not responding in time<sup>3 4</sup>.

Considering that the only possible changeable factor to reduce the number of accidents involving automated vehicles, is to improve the environment perception module. Sensing correctly everything in the driving scenarios is of vital importance and one of the main goals for autonomous driving that researchers are working with. This issue, when solved properly, will allow automated vehicles to pass from level 4 (high automation) to level 5 (full automation). For this reason, the perception module requires an enormous increase in robustness to be able to handle complex environments and situations not only in highways, but also in urban scenarios.

Furthermore, as the automated vehicle's navigation also relies on a priory digital map for environment perception of certain elements like speed limits, road geometry or driving directions; if the digital map is out of date and contains erroneous or missing information, the vehicle behavior might result in dangerous maneuvers. For the Vehicle Control module, this information is crucial in order to be able to calculate the right commands (acceleration/deceleration, steering angle) and control the lateral and longitudinal movements properly following as close as possible the chosen path (route). Therefore, if digital map is not available and neither the GPS information to localize the vehicle position, the perception module will need to provide sufficient information to the following modules to navigate a safe route.

The importance of road geometry, speed information and detection of other relevant agents in the environment, were the main motivations for the work in this thesis to perform path tracking. The dotted light blue blocks in Fig. 1.3 illustrate the modules we will focus on for contribution. Although most of the works in the literature make use of fusing sensory information [174], the work presented here will focus on the most available commercial sensors : camera for the environment perception and GPS receiver to locate and record the pathways to follow for the vehicle control module.

---

3. <https://www.digitaltrends.com/cool-tech/most-significant-self-driving-car-crashes/>

4. <https://www.theverge.com/2017/11/8/16626224/las-vegas-self-driving-shuttle-crash-accident-first-day>

## 1.3/ OBJECTIVES

With the aim of improving path tracking in urban scenarios using vision and GPS receiver as the main sensors, our research addresses the following topics :

- Vision-based environment perception through semantic segmentation.
- Visual object detection and classification of potential objects of interest like traffic signs.
- Path tracking by the means of vehicle control analyzing the road geometry and speed limits.

We focused on vision-based perception for the simple reason that vision is the main sense used by human drivers to perceive the environment. The final goal of its use, is to emulate it and if possible, improve it. Therefore, using cameras to capture the real world through images and applying computer vision techniques to analyze them, is crucial for autonomous driving technologies. GPS, on the other hand, provides global localization of the vehicle and is used for drawing the recorded path in a map to later use it for path tracking.

As mentioned earlier, the contributions of this thesis make reference to two modules of the autonomous vehicle (Fig. 1.3) : environment perception and vehicle control. We enumerate our objectives in the following :

1. Due to the lack of public information regarding French urban scenarios, the first objective is to construct the necessary datasets to work with. A dataset called UTBM-2 is proposed which includes visual and GPS information. The images are labeled in a per-pixel manner with 27 semantic classes and 10 instance ones. Such dataset is proposed for evaluating path tracking, semantic and instance segmentation approaches. A second dataset, comprising European traffic signs, is proposed to perform classification tasks. This dataset contemplates a standard common definition of traffic signs to categorize them according the Vienna Convention on Road Signs and Signals [1]. The purpose of this dataset was to deal with intraclass variability for European traffic signs and be able to classify them robustly after the perception module identifies them. In order to evaluate traffic sign detection in a well known benchmark considering all possible traffic signs in urban scenarios, an extended version of the German Traffic Sign Detection Benchmark (GTSDB) [83] was proposed. This third proposed dataset was labeled with masks of all the missing traffic signs. The three proposed datasets are important as they will be used

later to achieve other objectives.

2. The second objective is to perform accurate path tracking. This task required the evaluation of multiple control algorithms to find the advantages and drawbacks for each tested method. Based on the results, we proposed to include road geometry and speed information to improve lateral errors for the vehicle control based on GPS information. Our proposed inclusion module for the vehicle control system considers GPS information, creates a speed limits dataset and extracts sharp curved regions to perform path tracking.
3. Furthermore as the vision is necessary to perceive the environment, the third objective is to detect all objects of interest in the driving scenario. These objects include the road, lane markings, pedestrians, cyclists, traffic signs, traffic lights, crosswalks as well as cars, trucks and buses. In this thesis, we proposed a system that combines semantic segmentation approaches to detect the environment, with instance segmentation to localize individual instances of certain classes for further classification. The proposed system follows a strict data flow and filters to provide a good representation of the environment.
4. The fourth objective is to classify the individual instances of the traffic sign class. Here we proposed a Convolutional Neural Network architecture (CNN) to perform traffic sign classification which is included in the general environment perception system.

## 1.4/ THESIS ORGANIZATION

We will like to remark the reader that all the objectives mentioned above, were implemented not necessarily in the order they were described, but for comprehension reasons, we will present them following the organization given below.

Chapter 2 introduces the three proposed datasets constructed for French environment perception, path tracking, traffic sign detection and classification of text and symbol signs. Each dataset is used for evaluation in the rest of the thesis chapters.

In Chapter 3, we perform traffic sign classification with our proposed CNN architecture called Class\_CNN, and evaluated it with the proposed European traffic sign dataset. We introduce this contribution first, as vehicle control requires the visual recognition of speed limit signs and because this classifier will be included after the traffic sign detection is performed.

Chapter 4 is considered as the main heart of this thesis. Here we present a system proposal composed of two modules to perceive visually the objects of interest in the driving scenarios. The system is capable of segmenting the object regions (semantic segmentation module) while at the same time classifying individual traffic sign instances with the proposed CNN classifier presented in Chapter 3 (traffic sign recognition module). Each module is evaluated separately with the proposed UTBM-2 dataset, but their outputs are fused to provide a final image representation.

While the final contribution chapter dedicated to path tracking, Chapter 5, is presented at the end, its implementation was performed at the beginning of my research work. Towards a general overview of path tracking considering both GPS and visual information, the proposed control system is discussed at last. Nevertheless, due to lack of time, our contributions relate only to GPS signal analysis. The proposed vehicle control system handles road geometry and speed information to provide accurate path tracking respecting speed limits according to the French traffic zone rules.

Chapter 6 concludes this thesis and specifies the possible research directions to ameliorate this work.





## CONTRIBUTION





# PROPOSED DATASETS

## 2.1/ INTRODUCTION

Diverse and large scale datasets [106, 127, 116] have been proposed for supervised learning tasks in computer vision. Depending on the problem, there are different types of labeled datasets. However, regarding autonomous driving, the existing datasets are limited to certain scenes, sizes, complexity, annotation type and geographic distribution [45, 78, 79, 83, 135, 166].

As we will focus on scene understanding, particularly for French urban scenes, we require databases that fit this requirement. Apart from the Stereopolis dataset [52], which provides only information for traffic signs detection ; there is not many material to work with. To overcome such limitations, we proposed two datasets : one designed for classifying traffic signs and, another one for identifying every object in the scene (including drivable areas, static and moving objects). Additionally, a third dataset is proposed for the detection of German traffic signs. This last dataset was introduced for evaluation purposes and comparison with other approaches.

In this chapter, we explain how the datasets proposed are constructed with the goal to contribute in some research areas with new challenges, and to use them for evaluation in our work.

We will start describing a dataset for traffic sign classification (ETSD) to continue with another one created for semantic and instance segmentation approaches (UTBM-2). The aforementioned one is created with the purpose of categorizing every object in a French urban scene (environment perception). Lastly, we will introduce the extended version of the GTSDDB[83], a proposed dataset for traffic sign detection focused on 164 text and symbol traffic signs classes.



FIGURE 2.1 – Intra-class variability examples of European traffic signs.

## 2.2/ EUROPEAN TRAFFIC SIGNS DATASET (ETSD)

### 2.2.1/ INTRODUCTION

Traffic signs provide crucial visual information in order to understand the proper driving conditions [187]. For example, they inform about speed limits, drivable lanes, obstacles, temporary situations, roadway access, restrictive areas, etc. Reasons why they are designed to be easily detectable, recognizable and interpretable by humans [79]. Standard shapes, colors, pictographs and text are used to define their meaning.

Nevertheless and besides the efforts to standardize traffic signs [1], there exists inter and intra variability between countries and between classes for specific traffic signs. For example, the inter variability is mostly seen between countries that do not follow a common convention [77], while intra variability is perceptible among places which agreed to follow one. In Europe, the Convention on Road Signs and Signals [1] established the common sizes, shapes and colors to be used but allows each country to choose its own symbols and inscriptions. Fig. 2.1 illustrates some examples of intra class variability where it can be seen that symbols do not only vary between countries but also inside each of them. Regarding the last issue, Croatia and France (Fig. 2.1 second row) use 2 symbols for pedestrian crossing sign in Danger category while Belgium has speed limit signs with and without adding the Km inscription. Germany also uses 2 different symbols in the pass-right class which belongs to Mandatory category (Fig. 2.1 fourth row). At the same time, the background color in some categories can vary as defined in [1]. For example, Croatia uses the two possible colors (yellow and white) for danger and prohibitory signs (Fig. 2.1, first and third rows) while the other countries stick to only one.

Due to the needs of a standard and more complete definition of traffic signs in Europe, we

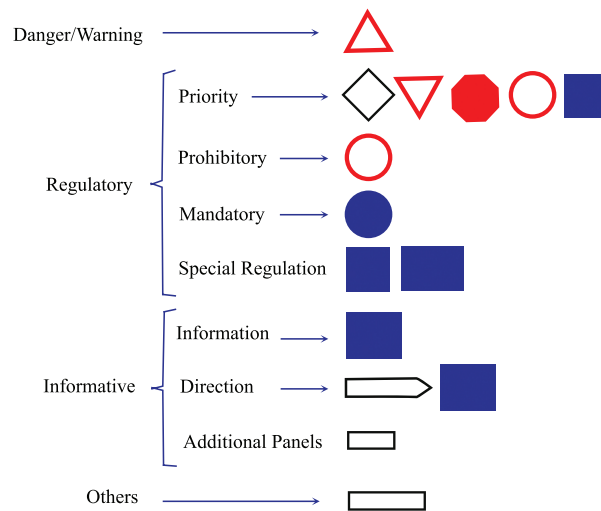


FIGURE 2.2 – European Traffic Sign Categories Definition. From left to right : main category, subcategories and most common shapes.

introduce a real-world dataset for traffic signs classification. The proposed ETSD dataset contains more than 80,000 images divided in 164 classes, which at the same time belong to 4 main categories following the Vienna Convention of Road Signs [1] (see Fig. 2.2). At the same time, our dataset deals with intra variability of classes (small variations of symbols or text for a class) which in our knowledge, not many studies have addressed.

The Dataset is composed of traffic signs from 6 European countries : Belgium, Croatia, France, Germany, Netherlands and Sweden. It gathers public available datasets and complements French traffic signs with sequences recorded in Belfort (France) and surroundings during Spring and Summer in 2014, 2015 and 2018. The sequences are composed of urban and rural environments and cover daytime and sunset conditions. The public datasets are composed of different scenarios (urban, rural, highway) mostly captured during daytime.

In the following subsections, a description of the standard data definition for the proposed dataset is provided to follow with, the acquisition procedure made to complement French traffic signs and be able to use the dataset for French urban scenarios. In the last subsection, we will describe the final composition of our proposed ETSD dataset.

### 2.2.2/ DATA DEFINITION

In order to standardize traffic signs, a lot of efforts have been made since 1909 to establish a common structure. Yet, in 1968 in Vienna, the United Nations Economic Commission for



FIGURE 2.3 – Examples of French directional signs.

Europe (UNECE) established the Convention on Road Signs and Signals which entered into force on 1978 [1]. Currently, 62 countries follow it with small variations in colors, pictographs and text.

The proposed dataset is divided into 4 relevant categories and subcategories (see Fig. 2.2) following the Vienna Convention on Road Signs and Signals [1] :

1. Danger warning signs. Warn road-users of a danger on the road and inform them of its' nature.
2. Regulatory signs. Inform road-users of special obligations, restrictions or prohibitions with which they must comply.
3. Informative signs. Guide road-users while they are traveling or provide them with other information which may be useful.
4. Others. Added class to inform road-users about important situations.

Regulatory and informative categories have subcategories that other datasets have considered for defining their traffic signs classes [79, 157]. For example, the German Traffic Sign Recognition Benchmark (GTSRB) [79] divided the traffic signs in 3 categories : 1) Danger, 2) Prohibitory, 3) Mandatory, and, include Other as a irrelevant category. Categories 2 and 3 belong to the Regulatory category defined in the Vienna Convention.

As we will explain in the following subsection, most of the public available datasets include only Danger or Mandatory signs. In the contrary, our proposed dataset considers a complete definition (including Informative signs) for traffic sign classification. The reason behind this, resides on facilitating the tedious task to recognize traffic signs which are not only composed of standard shapes with pictographs, but also to be able to recognize more complex signs for later interpretation. Directional signs are an example of that. They are composed with text and shapes to indicate certain direction (Fig. 2.3).

At the same time, additional panels provide complementary information to interpret traffic signs correctly. The broad selection of traffic signs resulted in 164 classes shown in Fig.

2.4.

A CSV file describing class names together with their categories and subcategories they belong to, is provided upon request together with the respective European dataset. In the following subsections (2.2.3 and 2.2.4), we provide more information about how our proposed dataset was built.



FIGURE 2.4 – Random representation of the 164 classes in the European traffic sign dataset.

### 2.2.3/ DATA COLLECTION - UTBM DATASET

We used the equipped vehicle of the UTBM laboratory to capture several sequences of images around university campus. The vehicle is equipped with multiple sensors (see Fig. 2.5) : a Bumblebee 3D stereo vision camera mounted on the top, a Real Time Kinematic (RTK) GPS sensor and several Light Detection and Ranging (LIDAR) devices. Data collection with the Bumblebee camera was performed with automatic exposure control and a frame rate of 16 fps. The sequences are captured during daytime covering urban environments with a resolution of 1280×960 pixels. Images for annotation were chosen every 3.5 meters.



FIGURE 2.5 – Equipped UTBM car.

In addition to the previous sequences, we captured images in the surroundings of Belfort with a conventional Canon Eos M camera with automatic exposure and focusing point. Driving scenarios cover urban and rural environments. The resolution of the video is 1920×1080 pixels at a frame rate of 24 fps. Images for annotation were chosen every 6 fps.

Data annotation was performed using the Training Image Labeler tool from MATLAB (see Fig. 2.6). Regions of interest of all seen traffic signs are labeled manually, cropped and saved in PPM file format.



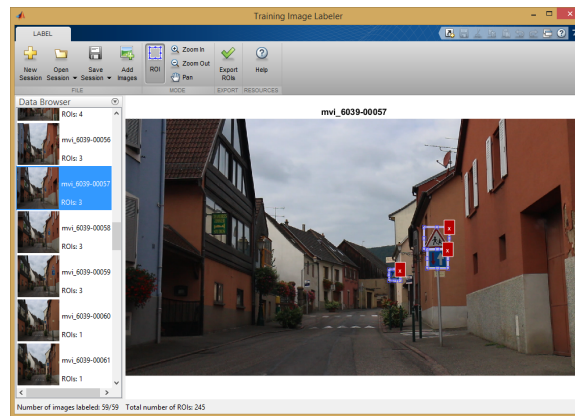


FIGURE 2.6 – Screenshot of the Matlab Training Image Labeler tool used for the manual annotation.

Every labeled sequence was recorded on a single tour in different days and places. Some sequences were recorded several times at different seasons and lighting conditions, this with the aim to include variance in the image dataset. All the recordings belong to France subset dataset.

Details about the number of images and traffic signs labeled will be provided in the next subsection, together with the other datasets.

#### 2.2.4/ DATA ORGANIZATION

ETSD gathers public available datasets from 6 countries : Belgium with the KUL Belgium Traffic Signs dataset [111], Croatia with MASTIF datasets [57], France with the Stereo-polis dataset [52, 78], Germany with the well-known German TSR Benchmark (GTSRB) [79], Netherlands with the RUG Traffic Sign Image Database [16] and Sweden with the Swedish Traffic Signs Dataset (STS dataset) [61].

In order to complement the already provided classes and add the missing ones (signs from Informative and Other categories), we labeled the German Traffic Signs Detection Benchmark (GTSDDB) [83] and the RUG [16] datasets.

A detailed description of the traffic signs per dataset is given below :

1. Belgium. KUL Belgium Traffic Signs dataset [111] is a dataset for classification where each image represents a sign with 10% offset. They are cropped according to the ground truth information to obtain only the Regions of Interests (ROIs). The original dataset is divided into 62 classes with 4,561 images for training and 2,528 images for testing. For the proposed ETSD dataset, its class 35 was divided into

3 classes (3 Mandatory signs), class 32 was divided into 6 classes (6 Prohibitory signs) ending up with 69 European classes.

2. Croatia. MASTIF datasets [57]. There are 3 datasets collected in different years (2009-2011). Each sign was annotated 4-5 times at different distances from the car. In order to distinguish images from each dataset, we put as prefix the dataset year for each image name.
  1. 2009. It was the richest classification dataset containing 6,423 signs already representing the Rols. This dataset is composed of 97 classes from which we divided "b31" class into 7 prohibitory classes, merged 19 classes into 6 and dropped 2, resulting in a total of 88 classes with 6,411 signs for our dataset.
  2. 2010. It is a detection dataset composed of 3,862 images of resolution 720×576 pixels. Signs were cropped according to the ground truth obtaining 5,184 Rols divided into 88 classes. We separated "A17" class into 2 dangerous classes, "b31" into 5 prohibitory classes, merged 18 classes into 4 and dropped 1 class. The total number of classes resulted in 68 in our dataset.
  3. 2011 dataset provides 1,013 images with the same resolution as in 2010 dataset. 1,429 traffic signs were cropped and distributed into 53 classes according to the ground truth. We divided "b31" class into 2 prohibitory classes, "c35" into 2 Special Regulation classes, "e19" into 2 Additional Information classes and merged 12 classes into 2. A total of 41 classes emerged for our European dataset.
3. France.
  1. Stereopolis dataset [52, 78] includes images acquired in Paris, France, grabbing a picture every 5 meters. It is made of 847 images of resolution 960×1080 pixels. We cropped the traffic signs with the ground truth provided and obtained 271 road signs divided into 10 classes. We split "a13a" class into 2 Danger classes and "b6a1" into 2 Prohibitory classes resulting in 12 classes for our dataset.
  2. UTBM dataset is composed of 86 classes with 2,631 signs. Image resolution varies as described in Section 2.2.3. A total of 1,863 images were labeled comprising 4 sequences captured with the Bumblebee camera and 6 sequences with the Canon Eos M.
4. Germany.

1. GTSRB [79] dataset contains a total of 51,839 images comprising 43 classes. The training and testing sets were used for our European dataset as their original definitions.
2. GTSDb [83] dataset comprises 900 full images with a resolution of 1360×1024 pixels. Originally 43 classes are labeled according to GTSRB, but we extended the dataset labeling manually with Rols the signs not considered in the image. We obtained 1,187 Rols composed of 46 additional classes. In total, we obtained 89 classes for our dataset (46 labeled classes + 43 original classes). We named these images with the prefix "GTSDb\_" followed by a unique index number.
5. Netherlands. RUG Dataset [16] contains 48 images of size 360×270 pixels in PNG format. Originally, the dataset was used to classify 3 classes (pedestrian crossing, compulsory for bikes and intersection). We labeled all traffic signs seen including the 3 classes as no ground truth is provided. 75 signs belonging to 12 classes were cropped and saved with the prefix "neth\_" followed by a unique number.
6. Sweden. STS Dataset [61] provides 2 sets (Set1 and Set2) with a total of 3,777 images labeled manually with Rols. 6,363 signs were extracted from the ground truth full images divided into 19 classes. We separated class "OTHER" into 34 classes. We named the new signs with the prefix "set1\_" and "set2\_" followed by a unique index for each corresponding set.

The naming convention used for each sign is as follows :  $X\_originalName.ppm$  Where  $X$  is replaced with a "B" for Belgium, "C" for Croatia, "F" for France, "G" for Germany, "N" for Netherlands and "S" for Sweden. We provide a CSV ground truth file to reference the given names with the original files in the necessary cases.

Traffic sign sizes vary between 6 and 780 pixels w.r.t the longer edge. Bounding boxes are not necessarily squared due to the nature of the traffic signs. Our final dataset takes care of intra class variability, as can be seen in Fig. 2.7, comprising 82,476 signs with 164 classes. We split the dataset into training and testing considering 70 - 30 % ratio only in the cases where the class contains at least 10 signs. If the class contains less than 10, we consider 100% for training. Fig. 2.8 illustrates the distribution by category of the classes with less than 10 signs. Moreover, we carefully selected random images per country for testing. In this way, we make sure the testing phase validates the accuracy for European signs and not for a specific country. The training and testing sets are ordered by class. Table 2.1 shows a summary of the classes and images considered for training and testing per country.



FIGURE 2.7 – Examples of some classes that contain intra class variability. Letters in the images are the initial of the country name they belong to. For example, B stands for Belgium, C for Croatia, etc.

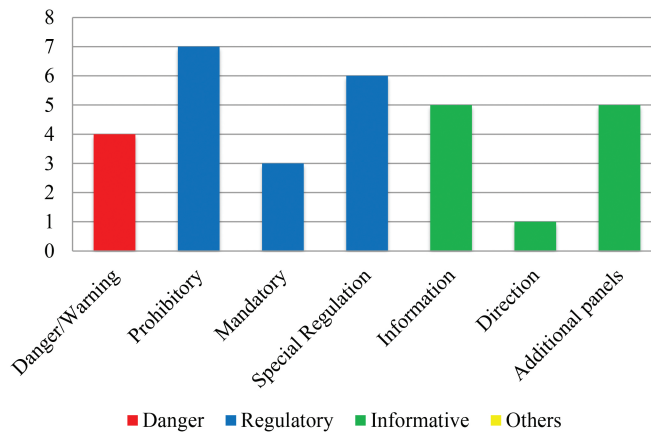


FIGURE 2.8 – Number of classes per category with less than 10 signs.

It is important to mention that even though the dataset is unbalanced as seen in Fig. 2.9 (with some classes containing less than 10 images), the purpose of its definition was to expand and establish a common notation for traffic signs classification based on the Vienna Convention [1]. The aforementioned will allow future researchers to contribute with more data in some categories or specific classes.

TABLE 2.1 – Relation of classes and total number of traffic signs by country.

Country	Dataset	Original classes	European classes	Training images	Testing images	Total images
Belgium	KUL	62	69	4561	2528	7089
Croatia	MASTIF-2009	97	88	4568	1843	6411
	MASTIF-2010	88	68	3694	1490	5184
	MASTIF-2011	53	41	1037	389	1426
France	Stereopolis	10	12	203	68	271
	UTBM	86	86	1878	753	2631
Germany	GTSRB	43	43	39209	12630	51839
	GTSDb	43	89	859	328	1187
Netherlands	RUG	3	12	58	17	75
Sweden	STS-Set1	19	52	2092	829	2921
	STS-Set2	19	52	2409	1033	3442

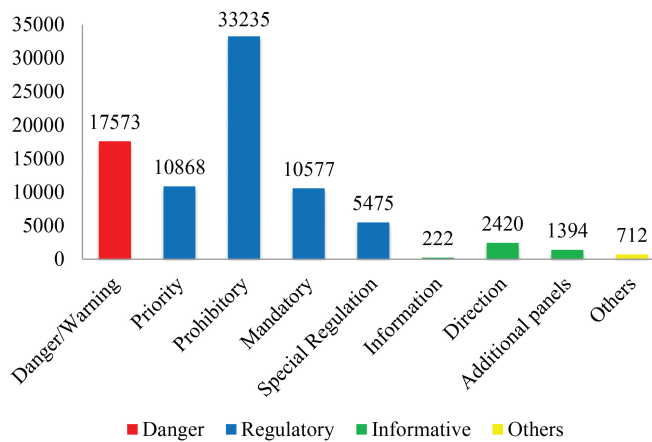


FIGURE 2.9 – Number of images grouped by category and sub-category of the proposed European dataset.

ETSD contains images that deal with occlusions, different lighting conditions, motion blur, human made artifacts and perspectives (see Fig. 2.10). Although, it does not contain signs captured during night or extreme weather conditions, data-augmentation is a well-known and applied technique in the literature for classification tasks to generate data and reduce the effect of over-fitting [158, 182]. Normally, the synthetic images are generated applying different transformations, simulating dark or bright scenarios, adding noise, blur, etc. to compensate the lack of information and avoid data collection and labeling. In this way, if data-augmentation is applied, we can say that our proposed dataset possesses the necessary characteristics as illustrated in Fig. 2.10 to consider it more exhaustive for

classification tasks.

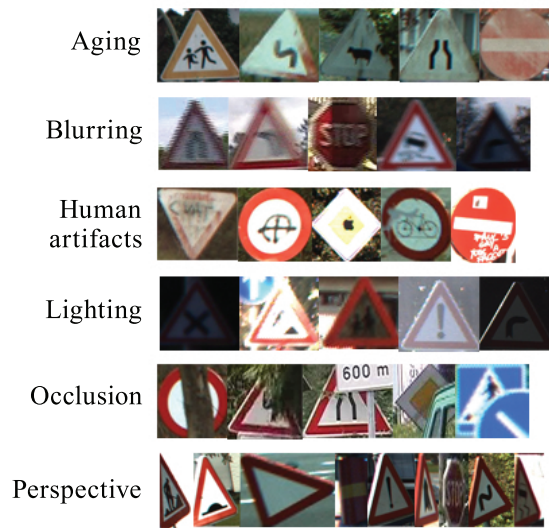


FIGURE 2.10 – Examples of challenging signs

### 2.2.5/ CONCLUSIONS

We proposed a traffic sign European dataset which deals with intra-class variability from 6 countries (Belgium, Croatia, France, Germany, Netherlands and Sweden). Such characteristic is a crucial aspect for Advanced Driver Assistant Systems (ADAS) or autonomous vehicles when driving from one country to another, since a classifier does not perform properly when traffic signs (pictographs or text) are slightly different from each other [158]. In Europe, this is a vital issue considering that countries are relatively close to each other. For this reason, defining a traffic sign dataset that contemplates the aforementioned problem, conducts our work to make an important contribution for intelligent vehicles.

Interestingly, no matter how many conditions our proposed European dataset considers (Fig. 2.10), there will always be hard situations for the classifiers to learn. In order to overcome this issue, image processing techniques can be used to enhance the visibility of an image and data-augmentation can be applied to improve the learning process generating more samples with different transformations.

The nature of Information category classes (classes based on text and with very different aspect ratios) makes them hard to learn since, as they contain text that varies depending on the country, the appearance is always different. This category was proposed in the dataset with the aim to allow other researchers to choose the signs they are interested in and provide data to solve other complex tasks like text recognition and interpretation.

At the same time, it can open new horizons to interpret correctly the traffic signs when additional panels (Informative subcategory) are located below a traffic sign.

## 2.3/ UTBM-2 - A SEMANTIC AND INSTANCE SEGMENTATION DATA-SET

### 2.3.1/ INTRODUCTION

Visual scene understanding is one of the main goals of ADAS or autonomous vehicles [119]. It requires visual data analysis in order to interpret the traffic agents and make the right driving decisions. This visual analysis requires the recognition of relevant scene object categories as well as locating and enumerating them. Most of the time, the scene understanding is performed through deep learning approaches which need a substantial amount of annotated data and computational resources to achieve it in real time [166].

The image scene analysis can be performed directly by detecting certain objects (object detection) and finding their locations with bounding boxes and their instances [99, 117, 126, 99, 143], categorizing each pixel in the scene (semantic segmentation) [175, 141, 123, 114], or a combination of both (instance semantic segmentation). The last approach detects individual objects and identifies the specific pixels which belong to the object class [162, 136].

Numerous efforts have been carried out in the design and creation of street level datasets to facilitate the complex visual understanding task. Some examples of such datasets are CamVid [45], Kitti Vision Benchmark Suite [82], Cityscapes [135], Leuven [35], Daimler Urban Segmentation [89] and Mapillary Vistas [166]. Nevertheless, some datasets are limited to the number of images, classes, geographic distribution, appearances and image size. As our objective is concerned with French urban scenarios to recognize specific classes like road, pedestrians, cars, traffic lights, traffic signs, lane markings and crosswalks ; we proposed a new ground truth dataset called UTBM-2 which allows quantitative evaluation for recognition, detection and segmentation approaches.

We first describe the specification of the dataset regarding the acquisition and characteristics. Then, the labeling procedure is explained to continue with the data division for training and testing sets. The dataset is constructed in different formats for compatibility with other datasets. In the last subsection, we summarize the characteristics and advantages of our UTBM-2 dataset.

### 2.3.2/ DATA SPECIFICATIONS

The UTBM-2 dataset belongs to a video sequence recorded by Qiao et al. [125] in Belfort, France. The sequence was captured with the equipped vehicle of the UTBM laboratory around university campus. Images were captured with Bumblebee stereo vision camera with automatic exposure control and a frame rate of 16 fps. The outputs are 3 RGB images corresponding to left, center and right cameras. Each 8 bit image was rectified using the estimated intrinsic parameters obtained with the method [11].

Additionally, every image is associated with a geographical position captured by the Real Time Kinematic (RTK) GPS sensor. In this way, a precise location of the images is stored for localization purposes.

UTBM-2 sequence has the following characteristics :

- It was captured during daytime representing typical French urban environments.
- Image resolution is 1280×960 pixels.
- Images for annotation were chosen from the left camera every 3.5 meters, resulting in a total of 541 images for UTBM-2 dataset.

### 2.3.3/ DATA LABELING

After a careful inspection of UTBM-2 images, we identified 27 classes of interest including fixed and moving objects. Classes were selected based on their relevance for applications as well as their compatibility with existing datasets [45, 135, 166]. We based the initial labeling on the Cityscapes dataset [135] as it was the only real world dataset in 2017 that contain lane markings on the road. The class names and their corresponding colors are given in Fig. 2.11. The void label makes reference to irrelevant objects. We used the color-index assignments following Brostow et al. [45] for the common classes and adding new colors for the extra 8 classes considered (last 2 rows of Fig. 2.11). For the new classes, we want to highlight the need of textTrafficSigns for its inclusion in recognition approaches, crosswalk to detect the hazardous zones where pedestrian are most likely to traverse, and refugeIsland to not drive where usually traffic lights and traffic signs are place on.

We labeled UTBM-2 dataset in a pixel-precision manner allowing accurate learning of appearance and shape. In other words, we labeled every pixel in the image with a corresponding class which is mostly used for semantic segmentation approaches. We used LIBLABEL [98], a tool written in MATLAB, to annotate object classes through polygons.



road	laneMkgsDriv	trafficLight	bus
tree	laneMkgsNonDriv	bicyclist	sky
building	parkingBlock	pedestrian	car
bridge	column_pole	sidewalk	void
fence	symbolTrafficSigns	vegetation	
bikeRack	textTrafficSigns	trafficSignBack	bicycle
crosswalk	pedestrianTrafficLight	refugeeIsland	trailer

FIGURE 2.11 – List of the 27 classes and their corresponding colors used for labeling.

Each polygon drawn is saved as an instance of the semantic class selected. In this way, it is possible to have an ID of the semantic class and an ID of the instance to simultaneously detect objects and segment them (instance segmentation). The annotation procedure was performed labeling objects from back to front to avoid non-labeled pixels and to define clear object boundaries. This procedure took approximately 316 hours of work for 541 images.

In order to provide a useful dataset for instance segmentation approaches, we chose 10 of the 27 classes to save their corresponding instances. These instance classes include moving objects (pedestrians, bicyclists, cars, buses, trucks, bicycles) as well as stationary objects (symbol-based traffic signs, text-based traffic signs, traffic lights, crosswalks). The decision of the stationary objects resides on the interest of : 1) detecting and classifying traffic signs, either all of them or only the symbol-based signs as covered in the GTSRB [79] and GTSDDB [83], 2) identify traffic lights for approaches that use them as the base for control algorithms (stop-red, go-green), 3) detect the closest pedestrian crosswalk marks on the road to be alert if moving objects are detected in/near the area.

Fig. 2.12 shows the distribution of the annotated pixels for each of the 27 classes grouped into 8 categories : flat, marking, construction, nature, vehicle, object, human and void. Traffic signs, lane markings, crosswalks, traffic lights, pedestrian, among others, are some of the less representative classes of the UTBM-2 dataset (see Fig. 2.12) due to the size they occupy in the image as well as the frequency they appear. Because of these characteristics, detection and recognition approaches have troubles identifying them properly.

The number of instances per class is illustrated in Fig. 2.13. Different from other datasets [45, 135, 166], we separated the traffic sign class into text and symbol classes, and provided their instances separately. In this way, if the text traffic signs are of interest for the researcher, they can be taken into account.

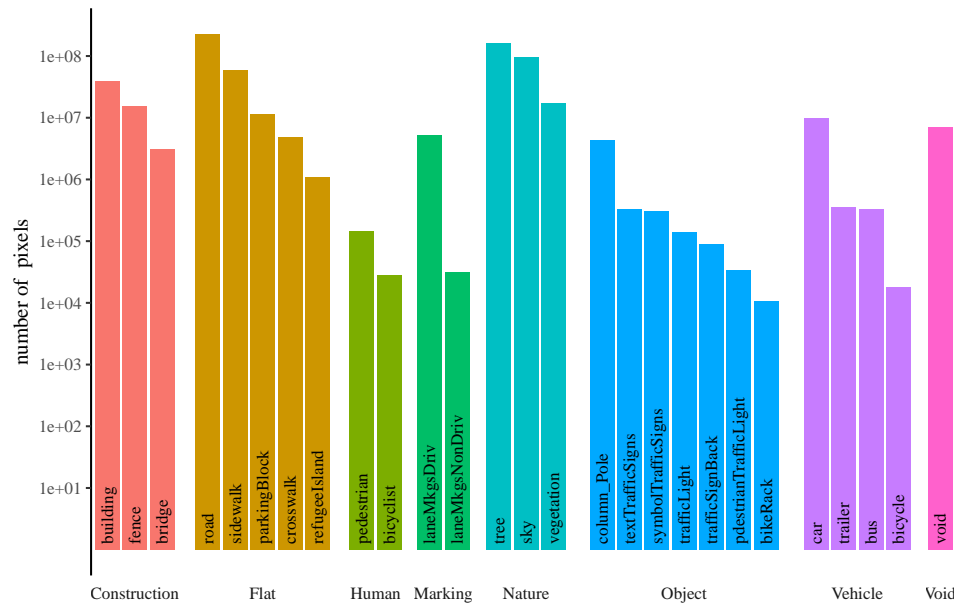


FIGURE 2.12 – Number of finely annotated pixels (y-axis) per class and their associated categories (x-axis).

### 2.3.4/ DATASET SPLITTING AND FORMATTING

Similar to Cityscapes [135] and Mapillary Vistas datasets [166], we decided to fix a ratio to split the dataset into training and testing sets. We divided it into 70%-10%-20% for train, validation and test sets equivalent to 378, 54 and 109 images respectively. Initially, we split the UTBM-2 dataset randomly and analyzed the data representation of the test set. If more than 5 consecutive images in the sequence are chosen, we discard 2 and chose others representing a different street scenario. In this way, the test set contains variability of scenarios.

Fig. 2.14 shows an example of a labeled image of UTBM-2 dataset together with its corresponding semantic and instance outputs. The dataset is converted into CamVid [45], Cityscapes [135] and Mapillary [166] formats to facilitate its usability depending on the goal task. Table 2.2 describes in general terms the UTBM-2 dataset characteristics together with its different formats.

Additionally, we labeled each traffic sign with its corresponding class based on 164 classes of the European traffic sign database proposed in our previous section 2.2. This dataset contains symbol and text based signs grouped into 4 main categories (danger, regulatory, informative and others) with very different shapes, colors and aspect ratios.

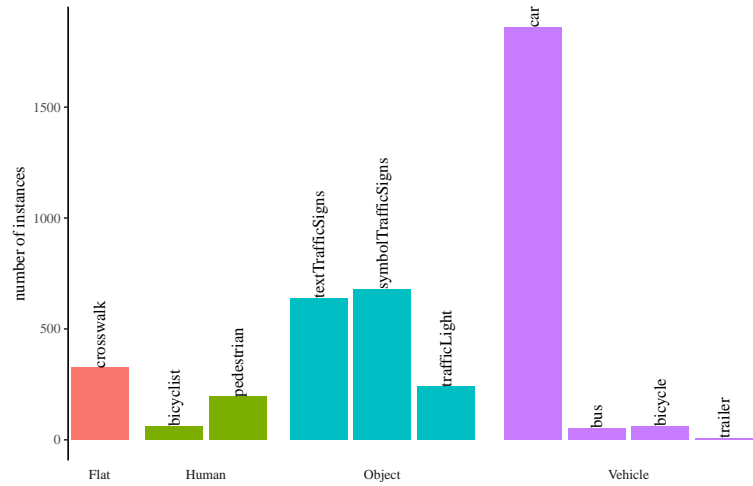


FIGURE 2.13 – Illustration of number of labeled instances per category and corresponding class.

TABLE 2.2 – Number of semantic and instance classes annotated in the UTBM-2 dataset and its corresponding conversion to different formats (CamVid, CityScapes, Mapillary). 'Original' makes reference to the number of classes that the original dataset format has, while 'Added' means the number of classes not included in the dataset but considered for annotation as extra ones.

Dataset	Semantic classes		Instance classes		# Images	Resolution (pixels)
	Original	Added	Original	Added		
UTBM-2	27	0	10	0	541	1280x960
UTBM-2 (CamVid)	32	8	0	10	541	1280x960
UTBM-2 (Cityscapes)	30	7	10	4	541	1280x960
UTBM-2 (Mapillary)	66	0	37	0	541	1280x960

We inspected each traffic sign labeled in the semantic images and discarded the signs which are not recognizable by human eye (normally very small and far signs). In the same way as the GTSDDB [83], we saved in a txt file the traffic sign class information together with the corresponding RoI. In case that, semantic segmentation approaches require the specific traffic sign label as part of the semantic classes, a multiplication of the RoI with its corresponding mask will allow to exchange the label classes.

### 2.3.5/ CONCLUSIONS

A dataset representing urban French scenarios was proposed for semantic, instance segmentation and detection approaches. Our UTBM-2 dataset is composed of 541 images

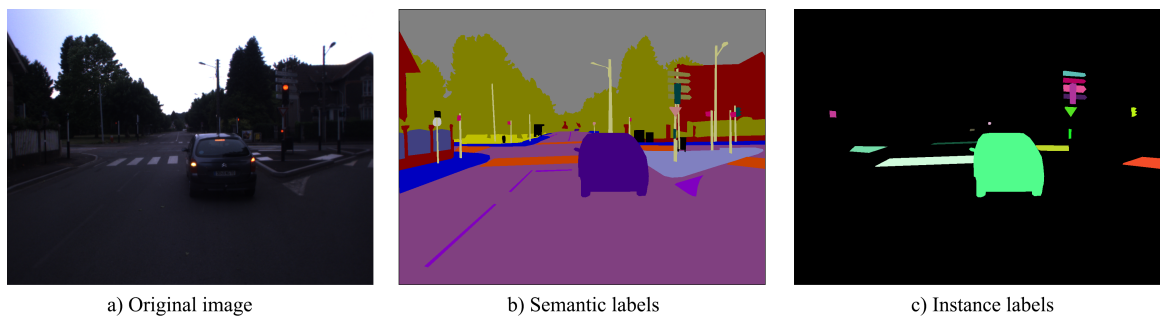


FIGURE 2.14 – Example of a captured image of UTBM-2 dataset and its corresponding semantic and instance labels. For the semantic classes refer to Fig. 2.11. Instance labels are numbered from 1 to  $n$  for each class but for visualization purposes they are presented with random colors.

with a resolution of  $1280 \times 960$ . It is labeled with 27 representative semantic classes for street scenarios from which 10 also include instance IDs.

Lane markings, crosswalks as well as parking blocks are important to consider for the environment perception since they provide information about the driving lanes, potential hazardous zones where pedestrians walk and possible non drivable areas designated to park even if they seem part of the driving road. At the same time, we distinguished between text and symbol based traffic signs to allow more flexibility when detecting certain types of objects. In the literature, most of the works focus on symbol signs but text signs are also important to interpret the environment specially when no GPS is available or road works deviate the traffic.

Additionally, the specific class for each traffic sign is provided contemplating the ETSD definition (164 classes). At the same time, temporal information can be taken into account, as the UTBM-2 dataset belongs to a video sequence. The precise geographical information of each image is also stored for vehicle localization and mapping approaches if required.

Even if our proposed dataset contains few images, the goal of our contribution was to have a suitable set of French urban scenarios to work with and be able to evaluate methods on our own data. In this way, the comparison results can be performed on public available datasets and on data focusing on our problem.

## 2.4/ EXTENDED GTSDB - TRAFFIC SIGN DETECTION DATASET

### 2.4.1/ INTRODUCTION

One of the challenging real world computer vision problems is traffic sign detection. It has been studied for a couple of decades to help humans respect the traffic rules and enhance driving safety. Its main applications reside in advance driver assistance systems (ADAS), mapping and more recently, in autonomous cars [171]. Nevertheless, due to the dynamic nature of the outside environments, this task remains difficult.

Several datasets around the world have been proposed in an attempt to solve this problem. Among the most popular ones are : the Netherlands RUG dataset [57], the French Stereopolis dataset [52], the US LISA traffic sign dataset [77], the Croatian MASTIF datasets [57], the Swedish Traffic Sign dataset (STS) [61], the German Traffic Sign Detection Benchmark (GTSDB) [83], the KUL Belgium Traffic Signs Dataset [111], the Chinese Traffic Sign Dataset (CTSD) [153] and its expanded version CSUST [169], and the Chinese Tsinghua-Tencent 100K benchmark [155]. Regardless of all the previous contributions, most of the datasets are labeled with only symbol traffic signs and a limited number of classes, discarding important information provided by text traffic signs. For example, directional signs, inform the driver which path to take to reach certain place while deviation signs, provide alternative routes if road construction works are taking place. Additional panels, composed of text and symbols, are small signs usually located below a symbol sign to specify the cases that apply for that signalization. Only the MASTIF dataset includes text signs but most of the state of the art works focused on the GTSDB [83] and the GTSRB [79] to evaluate their proposed traffic sign recognition systems.

Following the trend for evaluating traffic sign detection methods, we proposed an extended version of the GTSDB, labeling traffic signs not included in the dataset but visible and recognizable in the scenes. This version contains the same number of images with complete information about the text and symbol traffic signs. The extended GTSDB comprises 6 extra categories from its original version (only includes four : danger, prohibitory, mandatory and others) with 164 classes, following the definition from ETSD dataset described in section 2.2. Fig. 2.15 shows some examples of the traffic signs visible in the original GTSDB and not contemplated in their ground truth.

In the next subsection a description of the GTSDB is first presented followed by the data labeling procedure to end up with a summary of our contribution.

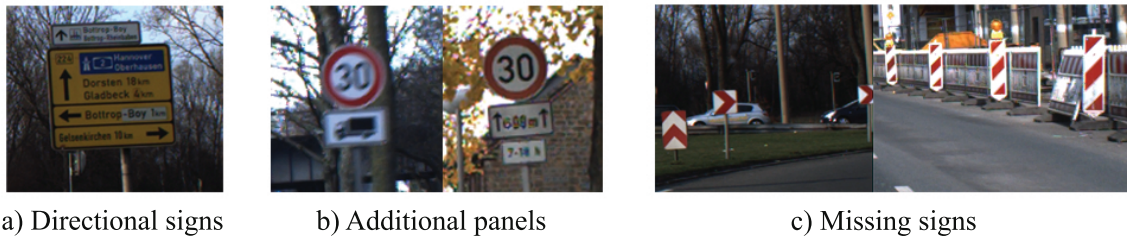


FIGURE 2.15 – Examples of traffic signs not included in the GTSRB. (a) shows four directional signs. (b) shows two examples containing additional panels. In the first example, the truck below the 30 speed limit sign, indicates that the restriction only applies for that type of vehicles; while in the second example of (b), the first panel announces that the speed limit is only applicable for 600 m and the second panel indicates the timing applicable. The missing signs (c) make reference to curves, obstacles, barriers and limited access on the side.

#### 2.4.2/ DATA DEFINITION

The GTSDDB dataset [83] introduced in 2013, is composed of 900 images of resolution  $1360 \times 1024$  pixels. The annotated traffic signs correspond to the 43 classes of the GTSRB [79] grouped into 4 main categories : prohibitive, mandatory, danger and other signs. The annotations are provided in a txt file with the RoI for each sign, together with its corresponding label class and the relations for each category. The dataset also includes labels of irrelevant classes considered in the category named 'other'. This last category was not included in their competition, but later works [181, 153] considered it for evaluation as well.

The training set consists of 600 images and 300 images correspond to the evaluation set. The traffic sign sizes vary between 16 and 128 pixels. Even though sizes seem relatively small, there are cases where smaller signs are visible and recognizable in the picture, but they were not labeled. Furthermore, there are many more important signs in the same scenario that were not considered. For this reason, we introduced the extended version of the GTSDDB which comprises signs belonging to the 9 sub-categories of the ETSD [184]. The ETSD comprises a broad selection of traffic signs including text-based signs as well as signs with very different aspect ratios. A more detailed description about this dataset is provided in section 2.2 and, some classification comparisons between methods using the ETSD dataset can be found in [184].

### 2.4.3/ DATA LABELING

Meletis and Dubbelman [183] labeled the masks for the 43 classes originally considered in the GTSDb and provided them to us. We labeled the GTSDb dataset with all the signs not included in the original dataset, but considered in the ETSD. The labels are provided in a pixel-precision manner to allow accurate learning of appearances and shapes. In other words, we labeled all the pixel in the image that correspond to a traffic sign, and the rest of pixels are labeled as background (2 classes - traffic sign and background). We used the LIBLABEL [98] tool in MATLAB, and annotated all the recognizable traffic signs with a polygon. In this way, we obtain a mask and instance for each sign to follow the same standard as in [183].

We inspected each traffic sing labeled and discarded the signs which were not recognizable by human eye (normally very small and far signs). The annotation procedure took approximately 75 hours of work for 900 images.

After the labeling and inspection were performed, the masks of the original signs [183] were combined with the masks of the missing signs (masks labeled). This merging procedure was performed carefully to preserve unique instance IDs for each sign. As the pixel labeling procedure contemplated a general traffic sign class and background, we used the Mapillary [166] format as a standard definition for the segmentation images. Fig. 2.16 shows an example of a labeled image of the GTSDb database together with its corresponding semantic and instance outputs.

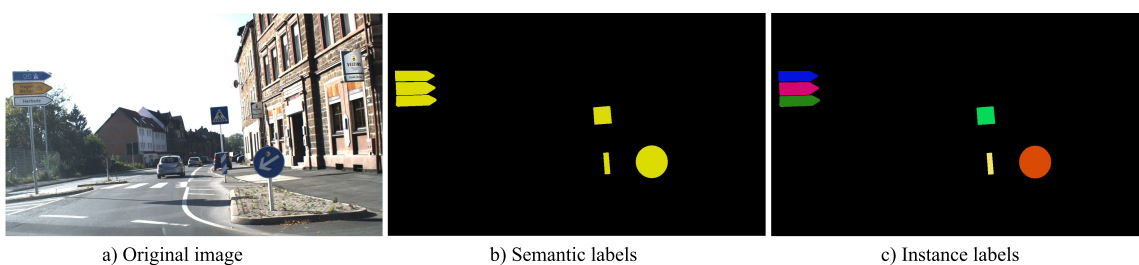


FIGURE 2.16 – Example of a GTSDb image and its corresponding semantic and instance labels. The semantic color class correspond to the Mapillary [166] definition. Instance labels are numbered from 1 to  $n$ , but for visualization purposes they are presented with random colors.

Additionally, each labeled traffic sign mask is saved with its corresponding class based on the 164 classes from the ETSD. The total number of traffic signs is expanded from 1213 to 2655 for the full dataset (900 images). Table 2.3 shows a description of the traffic signs by category of the original GTSDb database and its proposed extended version.

TABLE 2.3 – Number of signs by category for the training and testing sets of each dataset. The Others category in the original GTSDB and its extended version do not represent the same classes (refer to [83] for the classes in the original GTSDB and see Fig. ?? for the extended GTSDB) but we put it like that for the name convention. The unknown category contains traffic signs labeled which class does not correspond to any of the considered ones.

Category	Original GTSDB		Extended GTSDB	
	Train	Test	Train	Test
Danger	156	63	156	66
Priority	-	-	128	74
Prohibitory	396	161	478	193
Mandatory	114	49	122	56
Special regulation	-	-	115	73
Information	-	-	0	1
Direction	-	-	355	171
Additional panels	-	-	172	109
Others	186	88	284	92
Unknown	-	-	4	6
Total	852	361	1814	841

In the same way as in the GTSDB [83], a txt file containing the traffic sign Rol and class information is provided for detection approaches. In case that semantic segmentation approaches require the specific traffic sign label as part of the semantic classes, a multiplication of the Rol with its corresponding mask will allow to exchange the labels. The proposed extended GTSDB database can be found in <https://github.com/citlag/Traffic-Sign-Recognition>.

#### 2.4.4/ CONCLUSIONS

We proposed the extended version of the GTSDB which allows quantitative evaluation for recognition, detection and segmentation approaches. In other words, the dataset provides information about the Rol for each sign, as regular detection approaches need, but also a corresponding mask useful for segmentation and instance segmentation approaches.

The dataset is composed of 1814 traffic sign instances in the training set (962 signs more than in the original version) and 841 in the test set (480 added signs). It contemplates 164 symbol and text based classes grouped into 9 categories (danger, priority, prohibitory, mandatory, special regulation, informative, directional, additional panels and others) for detection approaches, and an extra category called 'unknown' for traffic signs not included in the ETSD.



Its definition was introduced with the aim of encouraging researches to contemplate the detection and recognition of all traffic signs. Thus, considering all the important and complementary information to interpret precisely the traffic rules. In this fashion, the intelligent vehicle system will be able to make the right decisions and pass them to the control agent to execute them. It is worth mentioning that, for complex driving scenarios with multiple visible signs, a hierarchy should be taken into account for priority rules.

## 2.5/ CONCLUSIONS

In this chapter, three datasets were proposed. The first one introduced to contemplate intra-variability of traffic signs between classes and European countries. Its 164 classes grouped into 9 categories-subcategories allow a exhaustive dataset definition for traffic sign classification approaches. The inclusion of text and symbol signs, give researchers the freedom to chose the classes they are interested in, and the possibility to contribute with more class variability. The introduction of this dataset allows multi-country traffic signs recognition which is an indispensable point for vehicles driving in a place different than where it was manufactured. Because in France, reaching one of the 6 neighboring countries is very common, autonomous vehicles need to take into account this vital issue for safety. In Chapter 3, we will analyze the importance of considering a complete traffic sign dataset evaluating it with a proposed CNN architecture and other state of the art methods.

The second dataset, UTBM-2, was introduced to evaluate environment perception in French urban scenarios. We contemplated 27 semantic classes from which we will focus on 9 to detect the road, moving objects, traffic signs, traffic lights and crosswalks. Due to the need of detecting separately instances for some objects, specially traffic signs, a proposed method for traffic sign recognition will be presented in Chapter 4.

Furthermore, the extended GTSDDB dataset was constructed with the intention of proving that the traffic sign recognition cannot be considered as a solved problem without contemplating all the traffic sign information encountered in an image scenario. The incorporation of regulatory, informative and other category signs, drops methods performance regarding traffic sign detection and classification. However, their information, as we will present in Chapter 3 and 4, needs to be considered for a proper interpretation of the traffic rules and to provide safety while driving.

# TRAFFIC SIGN CLASSIFICATION

## 3.1/ INTRODUCTION

As mentioned in Chapter 1, a part of the environment perception consists in identifying precisely traffic signs. This identification is carried out by computer vision systems through two steps : detection and classification. In this chapter we will focus on the classification part while the detection will be explored in Chapter 4.

Traffic sign recognition plays an important role for Intelligent Autonomous Vehicles and Advanced Driver Assistance Systems (ADAS). In the first case, the vehicle should be able to identify the traffic sign class to make a control decision similar to how humans do ; if the identification is wrong, or not perceivable, it will lead, in the worst case, to traffic accidents. For ADAS, the system helps the driver notice the traffic sign and, in consequence, ameliorate the driving route behavior.

Even though traffic sign classification seems to be an easy task and several research studies have claimed to achieve very high results [167], it still remains a challenging real-world computer vision problem due to the different and complex scenarios where traffic signs are placed into. For example, as traffic signs are found in outside environments, their appearances vary a lot due to illumination changes, weather conditions, aging and even the addition of some sticker or paint can make the traffic sign not easy to recognize. Moreover, considering that the vehicle is moving, other factor might affect the appearances like motion blur, not sharp image due to the camera focus, different perspectives and sometimes occlusions by other objects.

In an effort to deal with traffic sign recognition and because of the high industrial demand for autonomous vehicles, many studies have been published together with datasets from all over the world [16, 57, 61, 64, 78, 79, 111, 121, 77, 139]. Such systems, rely on either selected hand-coded features or the ones extracted automatically by learning. The

most effective ones, as proved in [79], rely on Convolutional Neural Networks (CNNs) architectures and are the ones in which we will focus on later in this Chapter.

Nevertheless, all approaches proposed to recognize traffic signs, are limited to a country and/or certain types of signs (shape, category). Being able to recognize the same traffic sign in different countries is still a problem that in our knowledge, not many studies have addressed, specially in a continent (Europe) where countries are a few hours apart.

In order to deal with all the previous constraints, a large number of sign examples should be considered to allow the system respond correctly when a traffic sign is encountered. For this reason, we will use our proposed European Traffic Sign Dataset (ETSD), described in Chapter 2, Section 2.2, to evaluate and compare several CNN architectures, among them our proposed one called *Class-CNN*.

We summarize our contributions in this chapter to the following :

- A proposed CNN architecture, composed of 5 blocks, capable of handling very robustly traffic sign classification.
- A comparative study of 6 CNN architectures trained with our proposed ETSD and the German Traffic Sign Recognition Benchmark (GTSRB) [79].

The rest of this chapter is organized as follows : Section 3.2 presents related work for traffic sign classification. Section 3.3 describes the neural networks architectures used for training with both datasets to continue with the description of our proposed *Class-CNN* in section 3.4. The analysis results are discussed in section 3.5, while conclusions and future work are presented in section 3.6.

## 3.2/ RELATED WORKS

The first work on traffic sign recognition was carried out in Japan in 1984 [24] and, since then, a broad number of works have been proposed to solve the problem through different techniques [158]. The most common ones are based on Support Vector Machines (SVM) [153, 51], template matching [7, 27, 56] and recently CNNs.

CNNs surpassed human performance on traffic sign classification [79, 158]. However, their architectures differ significantly from each other.

Even though traffic sign classification has been studied for decades, research works couldn't be compared until the German Traffic Sign Recognition Benchmark (GTSRB) [79] and the German Traffic Sign Detection Benchmark (GTSD) [83] were proposed.

Previously all research solutions have based their results on different public available datasets or on information acquired by their own.

The work of Abedin et al. [157] is an example of the formerly mentioned. They proposed the whole pipeline for detection and recognition. The recognition is carried out using SURF descriptors trained by an artificial neural network (ANN) with signs collected by themselves through video sequences in Bangladesh.

Islam et al. [163] performed classification on 10 Malaysian signs through an artificial neural network with a 2-layer feed-forward and a softmax classifier. Each class was composed of 100 samples dividing them into 70%-15%-15% for train, test and validation sets respectively. The signs were captured on roads and highways during different daytimes and weather conditions. Their dataset was also used by Lau et al. [121] to compare 2 classification methods, a Radial Basis Function Neural Network (RBFNN) and a CNN.

Li et al. [140] proposed a convolutional neural network (CNN) to detect and classify U.S speed limit signs [77]. Their network is based on a modified version of R-CNN [99] for the detection and a Cuda-convnet [47] for the classification. They claim to achieve 93.89% mean AUC [140] for 4 classes (No Turn, Speed Limit, Stop and Warning).

In the same manner, Jung et al. [139], collected and classified 6 types of traffic signs in South Korea. The training procedure was performed with LeNet-5 CNN architecture [9] predicting correctly 16 traffic signs on the road within an observable range.

Yang et al. [153] went beyond all the previously mentioned works, classifying not only the respective traffic sign classes but also their superclass (categories). Their system is based on 4-class SVM classifiers with RBF kernel using Color HOG features to detect the traffic sign categories. Then, three CNNs are used to perform real time traffic sign recognition. Each CNN contains two convolutional layers followed by sub-sampling layers, plus a fully-connected Multilayer Perceptron (MLP) in the last two layers. Their method was trained and evaluated with the GTSDB [79].

Aghdam et al. [158] designed a CNN architecture inspired by Ciseran et al. [68] and compared their work to 3 other networks [68, 90, 102] reducing by 65%, 63% and 54% the number of training parameters respectively. Their proposed CNN is trained with the GTSRB [79] and fine-tuned with the Belgium dataset [64, 111] in order to prove that their architecture is not only efficient, but also transferable.

Recent work proposed by Li and Wang [181] manages traffic sign detection and classification. Their traffic sign classification CNN uses different asymmetric kernels in order to

reduce the number of convolutional operations. Additionally, they fused different spatial information using an inception module by concatenating the output of 2 CNN branches along the channel axis. The CNN model was trained with the GTSRB proving to be effective and robust by obtaining 99.66% accuracy.

In our study, we will train different CNN architectures on the same datasets applying a common data preprocessing step and number of epochs in order to provide a fair comparison of traffic sign recognition approaches.

### 3.3/ CONVOLUTIONAL NEURAL NETWORKS FOR TRAFFIC SIGN CLASSIFICATION

It has been proven that CNNs are capable to solve problems with really high accuracy compared to human performance [118, 68]. Since the GTSRB [79], a lot of works were proposed to deal with traffic sign classification through different machine learning methods [68, 158, 139], from which CNNs outperform the others. As traffic sign classification is in high demand for the automotive industry, a lot of efforts have been made to achieve real time classification [68, 139].

We will describe 5 CNNs that achieve some of the best performances in the state of the art regarding Traffic Sign Classification.

#### 3.3.1/ LENET-5

LeCun et al. [9] proposed the well-known LeNet-5 convolutional neural network that is mostly used for handwritten recognition. Besides it was introduced in 1998, it became popular to solve other problems due to its simple and efficient architecture (see Fig. 3.1). It is composed of 7 layers, 3 Convolutional layers followed by Sub-sampling layers (except in the last), 1 Fully connected layer and the final output layer composed of Euclidean RBF units. The input size for this network is 32×32 pixels.

Jung et al. [139] used LeNet-5 to classify 6 types of Korean traffic signs obtaining an accuracy of 100% recognizing correctly 16 signs while driving on the KAIST campus road. As the results were promising in their study, we also trained the network with our proposed ETSD dataset for comparison.

### 3.3. CONVOLUTIONAL NEURAL NETWORKS FOR TRAFFIC SIGN CLASSIFICATION 43

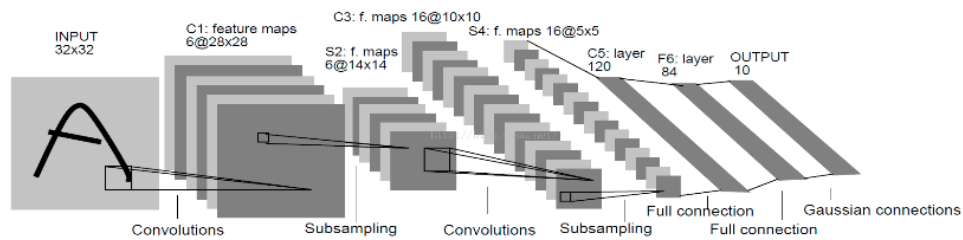


FIGURE 3.1 – LeNet-5 Network architecture proposed by Lecun et al. [9] for digits recognition.

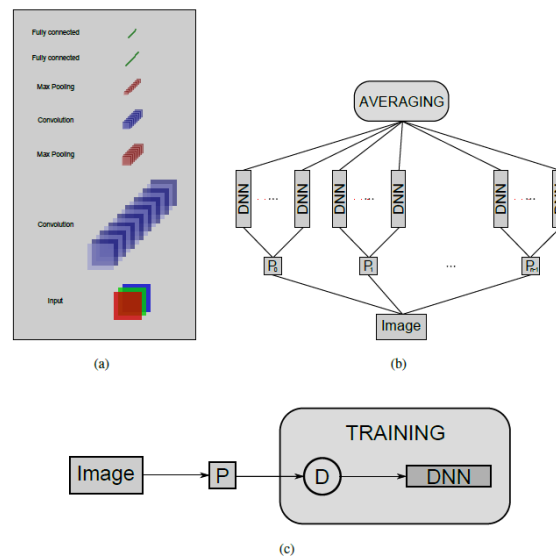


FIGURE 3.2 – Network architecture proposed by Ciseran et al. [68]. (a) DNN architecture. (b) MCDNN architecture. (c) Training a DNN. The dataset is preprocessed before training (P block), then, at the beginning of every epoch, the images are distorted (D block).

#### 3.3.2/ IDSIA MODEL

Ciseran et al. [68] based their work on combining several Deep convolutional Neural Networks (DNN) columns to form a Multi-column DNN (MCDNN). Their DNN network is composed of 2 Convolutional layers followed by Maxpooling layers. At the end, 2 fully connected hidden layers are used to pass the output to a final fully connected layer with 6 neurons to perform classification. They used a scaled hyperbolic tangent activation function for convolutional and fully connected layers. Their net takes as input 2 images of  $48 \times 48$  pixels and performs some distortions in each column to average at the end the final predictions of each DNN. The MCDNN architecture is illustrated in Fig. 3.2.

Aghdam et al. [158] trained this model with the GTSRB dataset and obtained an accuracy of 98.52% performing data-augmentation for the training set. In our study, we will train and evaluate their DNN architecture without considering an ensemble (MCDNN). This in order

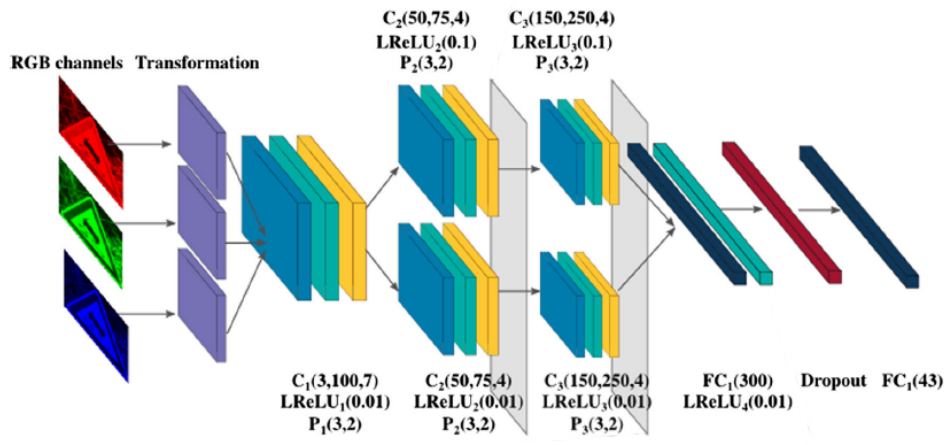


FIGURE 3.3 – Network architecture proposed by Aghdam et al. [158]. Light blue represents the convolution layers, green the ReLU activation layers, yellow the pooling layers, dark blue the fully connected layers and red the dropout layer

to compare the capability of their proposed DNN.

### 3.3.3/ URV MODEL

Aghdam et al. [158] made a comparative study between methods using the GTSRB [79]. Their CNN based on Ciseran et al. work [68] demonstrated, in their results, that their network is able to reduce complexity and computational time, improving accuracy compared to the one of Ciseran et al. Their Network is based on 3 convolution-pooling layers and 2 fully connected layers with a dropout layer in between to avoid over-fitting. ReLU activations [54] after each convolutional layer and after the first fully-connected layer are applied. Their network takes as input a  $48 \times 48$  RGB image and classifies it into one of the 43 traffic sign classes of the GTSRB dataset. They claim to achieve 98.94% accuracy performing data-augmentation for the dataset. Fig. 3.3 shows their network architecture.

### 3.3.4/ CNN WITH ASYMMETRIC KERNELS

Li and Wang [181] based their CNN design on convolutional layers using asymmetric kernel sizes to replace the usual symmetric  $n \times n$  kernel (e.g.  $3 \times 3, 5 \times 5, 7 \times 7$ ), with asymmetric ones defined by  $n \times 1$  and  $1 \times n$  in some convolutional layers. This replacement decreases the number of convolutional operations making the network more efficient. Their CNN architecture is composed of 3 convolutions with symmetric kernels, 6 convolutional layers with asymmetric ones ( $7 \times 1, 1 \times 7, 1 \times 3, 3 \times 1, 1 \times 7$  and  $7 \times 1$ ), and 2 fully connected layers. Each of these layers except for the last one (Softmax classifier) are followed by

### 3.3. CONVOLUTIONAL NEURAL NETWORKS FOR TRAFFIC SIGN CLASSIFICATION 45

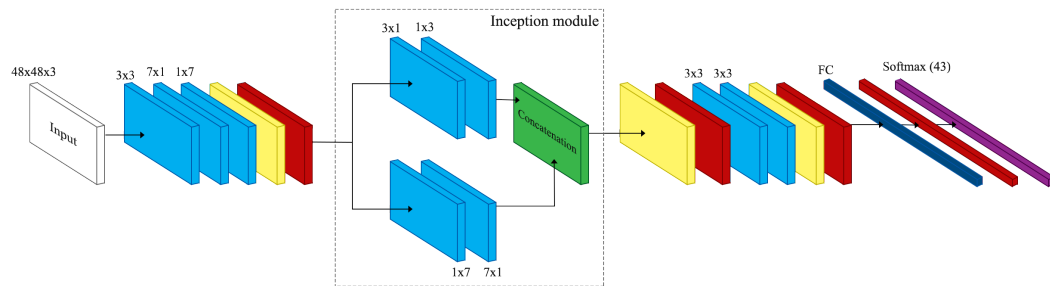


FIGURE 3.4 – CNN architecture drawn from [181]. Light blue blocks represent the Convolution + BatchNormalization + ReLu layers, yellow the pooling layers, red the dropout layer, green in this case refers to layer concatenation, dark blue shows a fully-connected layer + BatchNormalization + ReLu while purple refers to dense layer with Softmax activation. The numbers indicated for the Convolution layers refer to the kernel sizes used.

Batch Normalization [120] and ReLu activations [54]. They used an inception module with asymmetric kernels after the third convolution to learn different spatial information. The last two convolutional layers use symmetric kernels. Dropout technique [110] is used by the authors to avoid over-fitting. As they trained the network with the GTSRB, the output is set to 43 and input size to  $48 \times 48 \times 3$  (RGB image). The architecture of their CNN is illustrated in Fig. 3.4.

The performance of this CNN achieved 99.66% accuracy on the GTSRB test set trained with data-augmentation for 200 epochs. Despite the use of asymmetric kernels to decrease the complexity of the network, the number of parameters for this architecture is still high, compared to the others we will study here. A comparison study including the number of parameters will be provided in section 3.5.

#### 3.3.5/ CNN 8-LAYERS

Besides the previously mentioned architectures, we decided to train a classifier that does not represent a really deep network but competes with the state of the art. Networks composed with deep architectures (several hidden layers) have proven to provide the best results (Inception [128], VGG16 [109], ResNet[138]), but their complexity kills the computational time. For this reason, simple networks are used considering information preprocessing and data-augmentation [108] if the dataset does not contain enough examples for the learning phase.

Chilamkurthy [159] worked on the traffic sign image classification problem. Even though he did not mention in which Network he based his work, we could see that his proposal is like a VGG architecture [109]. There are blocks of Convolutional layers, activated by ReLu



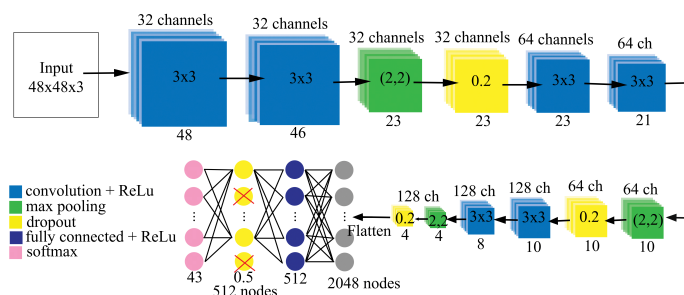


FIGURE 3.5 – Architecture drawn from Chilamkurthy proposal [159].

function [54], followed by Max Pooling and additionally Dropout layers. His architecture can be seen in Fig. 3.5. Different from VGG architecture, he added dropout layers with a range of 0.2 after each block of convolutional layers and, in the same way, after the fully connected layer with a range of 0.5. Dropout layers are used to prevent over-fitting and to make the network learn robustly its parameters [110].

Chilamkurthy reported 97.92% and 98.29% accuracies without and with data-augmentation respectively on the GTSRB test set. His network takes RGB images of size  $48 \times 48$  pixels as input, while transforming them to HSV color space and performing histogram equalization in the V channel.

A comparison of all the previously mentioned architectures will be performed in section 3.5 using the GTSRB and our European dataset.

### 3.4/ PROPOSED CNN (CLASS\_CNN)

After referring to several architectures [63, 68, 102, 153, 158, 181, 182], we defined a CNN based on 5 convolutional blocks to identify the specific traffic sign class. We call it Class\_CNN and its architecture is defined in Table 3.1.

Our proposed architecture is inspired by VGG-16 network [109] setting up blocks of convolutional layers with kernel sizes of  $3 \times 3$ . In an attempt to reduce the number of parameters and convolutional operations, we performed the same strategy as Li and Wang [181] did, and replace the  $n \times n$  kernel by a  $n \times 1$  and  $1 \times n$  kernels [150]. Using asymmetric convolutions, the computational cost decreases certain percentage calculated with  $(n \times n - 1 \times n - n \times 1) / (n \times n)$ . For example, replacing a  $5 \times 5$  kernel with asymmetric ones will save  $(5 \times 5 - 1 \times 5 - 5 \times 1) / (5 \times 5) \approx 60\%$  of computational cost, while for a  $3 \times 3$  will decrease by 33%. Authors in [150] claimed that this replacement doesn't work good in early layers but gives good results applied to feature maps of sizes between 12 and 20. We applied

TABLE 3.1 – Architecture of our CNN for traffic sign class identification (Class\_CNN). 'chan' makes reference to the number of channels, 'kS' to kernel size, 'std' to stride and 'kpProb' to the probability of keeping the original feature.

	type	outMap	chan	kS	std	kpProb
I	input image	48 × 48	3	-	-	-
B1	convolution	48 × 48	32	3 × 3	1	-
	convolution	48 × 48	32	3 × 3	1	-
	maxpooling	24 × 24	32	2 × 2	2	-
	dropblock	24 × 24	32	3 × 3	-	0.8
B2	convolution	24 × 24	64	3 × 3	1	-
	convolution	24 × 24	64	3 × 3	1	-
	maxpooling	12 × 12	64	2 × 2	2	-
	dropblock	12 × 12	64	3 × 3	-	0.8
B3	convolution	12 × 12	128	3 × 3	1	-
	convolution	12 × 12	128	3 × 3	1	-
	maxpooling	6 × 6	128	2 × 2	2	-
	dropblock	6 × 6	128	3 × 3	-	0.75
B4	convolution	6 × 6	256	5 × 1	1	-
	convolution	6 × 6	256	1 × 5	1	-
	dropblock	6 × 6	256	3 × 3	-	0.75
B5	convolution	6 × 6	320	3 × 1	1	-
	convolution	6 × 6	320	1 × 3	1	-
	maxpooling	3 × 3	320	2 × 2	2	-
	dropblock	3 × 3	320	3 × 3	-	0.75
C	fully connected	1 × 1	256	-	-	-
	dropout	1 × 1	256	-	-	0.5
	fully connected	1 × 1	43	-	-	-

asymmetric convolutions in blocks 2 and 3 (B2 and B3) but it only made worse the results. In our case, applying them in blocks 4 and 5 (B4 and B5) gave us the best outcome.

Differently from the state of the art, we applied 2 types of regularization techniques : dropout [110] and dropblock [178]. Dropout excludes randomly activations in the feature maps, keeping only certain percentage. It helps the model learn more robustly the parameters preventing overfitting. Dropblock, on the other hand, discards information of a continuous region (defined by a block of size  $n \times n$ ) of the map. In this way, neurons with similar features are discarded and propagated through the whole network changing every epoch and allowing it to generalize better (see Fig. 3.6). In consequence, the model tends to be more stable during learning as it forces itself to learn from the remaining activations in order to classify the input image. Because dropblock applies two dimensional kernels, it can only be used for convolutional operations whose feature maps have at least the same

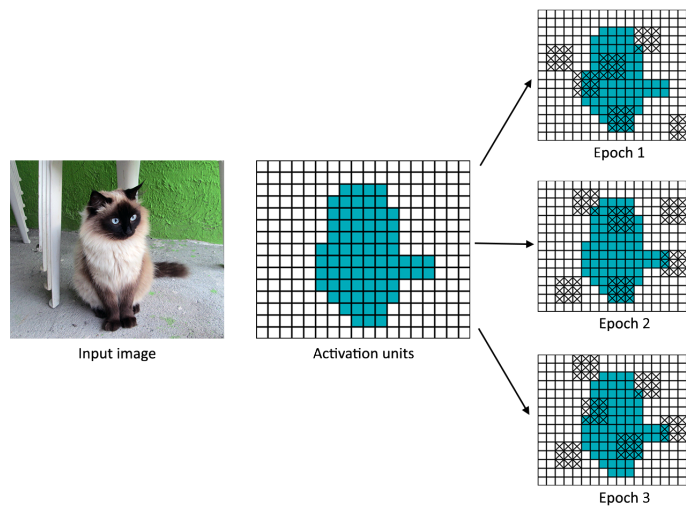


FIGURE 3.6 – Dropblock processing pipeline. The blue regions in the image representing the activation units, contain semantic information. Dropping continuous regions removes closely related information and consequently enforce remaining units to learn features every epoch for classifying the input image.

size as the region block. We performed experiments using both regularization techniques (see Section 3.5) and found that dropblock improves our model performance.

Additionally, Batch Normalization [120] (BN) is applied after each convolution to help the model learn more easily its parameters. ReLU [54] activations are used after the BN layer for each convolutional layer and fully connected layer, except for the last one, which uses a Softmax activation to output certain number of probabilities for each input. The output number is set to 43 in Table 3.1 due to the number of classes that GTSRB [79] has. This output changes depending on the desired number of classes.

For comparison purposes and in order to evaluate our proposed CNN, we trained the classifier (Table 3.1) with 43 output classes according to the GTSRB and with 164 outputs on the ETSD [184]. The experimental results are provided in the next Section.

### 3.5/ EXPERIMENTAL RESULTS

In order to perform a fair comparison between different networks, we trained the models described in section 3.3 and our proposed Class\_CNN with the GTSRB and the ETSD. All models are trained in GPU mode using a NVIDIA GeForce GTX1080Ti with 11GB of memory, an IntelCore i7K-8700K (6 cores 12 threads, 12 Mb cache memory) processor and RAM of 32GB. The learning process varied according to the complexity of each

TABLE 3.2 – Accuracy percentage results obtained on the GTSRB and European test sets. The input size refers to image "width×height×channels", while the number of parameters is presented in millions (M) and time in milliseconds (ms).

Model	Input size	GTSRB		European		Time
		Parameters	Accuracy	Parameters	Accuracy	
LeNet-5	32x32x1	0.13 M	89.1%	0.35 M	89.8%	0.0067 ms
IDSIA	48x48x3	1.54 M	94.62%	1.58 M	95.82%	0.6 ms
URV	48x48x3	1.12 M	96.1%	1.16 M	96.53%	0.61 ms
CNN_asymmetricK	48x48x3	2.92 M	97.88%	2.95 M	98.48%	0.39 ms
Improved_CNN_8-layers	48x48x3	1.48 M	98.52%	1.51 M	97.88%	0.15 ms
Class_CNN	48x48x3	2.09 M	98.51%	2.12 M	98.25%	0.16 ms

model.

### 3.5.1/ PERFORMANCE COMPARISONS WITHOUT DATA AUGMENTATION

The URV model proposed by Aghdam et al. [158] and the IDSIA model proposed by Ciseran et al. [68] are implemented in the Caffe framework<sup>1</sup>. The input image of both models is an RGB image of 48×48 pixels. We trained both models with the original parameters as stated by Aghdam et al. [158] changing only the batch size from 100 to 128 and the number of iterations to make it learn for the equivalent of 40 epochs (11500 iterations for the GTSRB and 17500 for the European dataset). In the same way, the test iterations are modified according to the validation dataset : 31 for the GTSRB and 48 for the European one.

Towards a fair comparison, we normalize the European dataset subtracting the mean image like it is done for the GTSRB dataset. The results presented in [158], based their accuracy on augmented-data carried out with 12 transformations (see paper [158] for more details).

UVR and IDSIA models were trained without performing any data-augmentation or pre-processing. This with the aim to evaluate the performance of the pure models. We use 10% of the training sets for validation (3921 images for GTSRB and 6055 for European dataset). The results obtained can be seen in Table 3.2, where the classification accuracies presented come from evaluating the models on the test sets.

In the same manner, we trained the models : LeNet-5 [9], the CNN with asymmetric kernels proposed by Li and Wang [181] (CNN\_asymmetricK), and the model proposed by

1. <https://github.com/pcnn/traffic-sign-recognition>

Chilamkurthy [159] implementing some changes to increase the model accuracy (Improved\_CNN\_8-layers). All these architectures are implemented in the Tensorflow framework.

Due to the nature of LeNet-5 [9], images are converted into gray-scale and resized to  $32 \times 32$  pixels. We set the batch size to 128 and the number of epochs to 40 to train on both datasets. The rest of the training parameters are left unchanged according to the original implementation<sup>2</sup>.

Since the outputs of LeNet-5 fully connected (FC) layers are reduced from 400 (total number of neurons after the pooling layer of the second convolution) to 120-84-N (output classes), we had to modify the number of neurons of the FC layers in order to be able to classify 164 classes for our proposed ETSD. For that, we changed the first output of the first FC to 300 and the second one to 200. In this way, we are able to classify 164 classes as the desired output. Results are available in Table 3.2.

Furthermore the model of Li and Wang [181] was trained as well for 40 epochs on both datasets. Images are resized to  $48 \times 48$  pixels keeping the 3 color channels (RGB). Different from the authors in [181], we applied a preprocessing step converting the image to HSV color space and equalizing the V channel. Preprocessing methods are used to normalize the image and give better contrast [102]. The mean image was also subtracted in each dataset. We used the model parameters as proposed in [181] and the results obtained are shown in Table 3.2. We refer to this model as CNN\_asymmetricK.

Moreover, and in order to improve the accuracy of 97.92% reported by Chilamkurthy on the GTSRB, the CNN 8-layers model [159] was modified adding 1) L2 regularization on each convolutional and fully connected layers and, 2) Batch Normalization after each convolutional layer and before the ReLu activations. The modified CNN is refereed as Improved\_CNN\_8-layers. The effect of Batch Normalization [120] has proven to make the network learn robustly normalizing the input parameters in each batch at each layer, to reduce their covariance shift. Regularization, in the other hand, helps reduce over-fitting adding a penalty in the loss function to combat high variance. L2 regularization was added with a value of  $1e-4$ . The optimizer was changed from Stochastic Gradient Descent (SGD) to Adam [104] with an initial learning rate of  $1e-3$  and a regularization coefficient of  $1e-3$ . Batch size was set to 128 and the input to  $48 \times 48$  pixels with 3 channels (RGB image). In the same way as with CNN\_asymmetricK model, the input image is preprocessed as mentioned earlier. The model was trained for 40 epochs validating the training process using the validation set (10% of the training set) which stops the training when over-

---

2. <https://github.com/sujaybabruwad/LeNet-in-Tensorflow>

fitting occurs (validation loss starts increasing and validation accuracy start decreasing). In this way, we make sure the CNN model learns correctly. For example, while training the Improved\_CNN\_8-layers model with the ETSD, it stops learning at epoch 16 obtaining a testing accuracy of 98.52% (see Table 3.2).

The changes implemented in the CNN 8-layers (Improved\_CNN-8layers) model made the network improved its accuracy by 0.6% in total, from 97.92% reported by Chilamkurthy [159] to 98.52% on the GTSRB dataset without performing data-augmentation. Besides the accuracy improvement, and due to the addition of Batch Normalization (BN) layers, the number of parameters also increased from 1.36 to 1.48 Millions in reference to the GTSRB dataset (43 outputs). Nevertheless and regardless the increase of complexity, the Improved\_CNN\_8-layers model is still competitive in accuracy and time compared to the others reported in Table 3.2.

Regarding our proposed Class\_CNN architecture, we trained it initially for 40 epochs using Adam optimizer with a learning rate of  $1e-3$  and a weight decay of  $1e-6$ . Additionally, we monitored the validation loss with a patience of 20 and factor of 0.2 to reduce the learning rate when the validation loss stops decreasing. In this way, only the best weights are saved preventing over-fitting. The RGB images are resized to  $48 \times 48$ , converted to HSV color space, and converted back to RGB after equalizing the V channel. An accuracy of 98.51% was obtained for the GTSRB while 98.25% resulted for the ETSD (see Table 3.2).

The accuracies obtained on both datasets (Table 3.2) are similar, varying from 0.26% - 1.2% between the GTSRB and ETSD, no matter the model used. Hence we can say that models are stable and results depend on the dataset itself.

The processing time for each model depends on its number of parameters and the framework used. For instance, the processing times presented in Table 3.2 are computed to predict the traffic sign class of a single image in GPU mode. Essentially, we can see that the models implemented in the Caffe framework (IDISA and URV) are relatively slower than the ones implemented in Tensorflow. For example, the IDSIA model (Caffe) has 1.54 Million parameters and takes 0.6 milliseconds to make a prediction, while CNN\_asymmetricK model (Tensorflow) has 2.92 Million parameters and takes 0.39 milliseconds (around 40% faster).

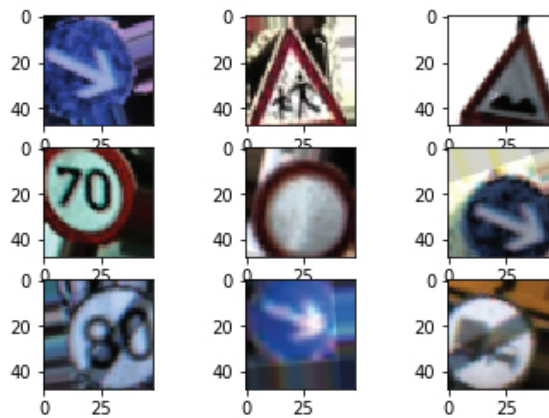


FIGURE 3.7 – An example of some augmented traffic signs from the European dataset.

### 3.5.2/ PERFORMANCE COMPARISONS WITH DATA AUGMENTATION

As mentioned before, techniques like data-augmentation also help to improve the accuracy of a classifier without acquiring and labeling more data. Therefore, we applied this technique with the 3 models which obtained the best accuracies on the test datasets : the CNN\_asymmetricK model [181], the Improved\_CNN-8 layers model and our proposed Class\_CNN. Luckily, as these models are implemented in Keras using Tensorflow as back-end ; Keras provides an option to perform real-time data-augmentation with its ImageDataGenerator class. We considered the following 5 transformations :

1. Width shift = +-4 pixels
2. Height shift = +-4 pixels
3. Scaling = [0.8,1.2]
4. Shear = [0,0.1] radians
5. Rotation = +-10 degrees

Besides that transformations, histogram equalization is also considered as data preprocessing. For this, the exact same procedure is applied as stated by the author in [159]. Some examples of augmented images can be seen in Fig. 3.7.

Considering that the models were previously trained without data-augmentation, we used their pre-trained weights as initializers for the new training procedures. This technique is also called transfer learning and, avoids learning everything from scratch. Normally, it is more common to use it with deep architectures which were trained on huge amount of data to adapt the model to a new output with less training examples [55]. In our case, we used it as initialization for the architectures to continue learning with the new generated

TABLE 3.3 – Accuracy percentage results obtained on the GTSRB test set. The input size refers to image "width×height×channels", while the number of parameters is presented in millions (M) and time in milliseconds (ms).

CNNs	Input size	Data augmentation	Parameters	Accuracy (dropout)	Accuracy (dropblock)	Time
Class_CNN_Symmetric	48x48x3	Yes	3.59 M	99.19%	99.54%	0.2 ms
Class_CNN	48x48x3	Yes	2.09 M	99.41%	99.51%	0.16 ms
Class_CNN_RB	48x48x3	Yes	2.40 M	99.08%	99.14%	0.17 ms

data. The training parameters were left unchanged as defined previously and only the number of epochs was set to 50.

### 3.5.2.1/ PROPOSED CLASS\_CNN

Before comparing the three CNNs chosen for training with data-augmentation, we will first demonstrate the performance of our proposed Class\_CNN defined in Table 3.1, with self comparisons of the architecture adding the following modifications :

1. A classifier that changes the asymmetric kernels for symmetric ones of size  $3 \times 3$  (Class\_CNN\_Symmetric).
2. A classifier which adds 2 residual blocks (Class\_CNN\_RB), one between B3 and B4 (see Table 3.1) and one between B4 and B5. The first residual block uses a convolutional layer with 128 filters and kernel size  $1 \times 1$  and the second one a convolutional layer with 64 filters and size  $3 \times 3$ . Both convolutions in the residual blocks are followed by BN and ReLu, and are concatenated according to the ResNet [138] definition.
3. The 3 CNNs (Class\_CNN, Class\_CNN\_Symmetric and Class\_CNN\_RB) defined only with dropout as regularization.

For a visual representation of our Class\_CNN modifications, refer to Fig. 3.8. Table 3.3 shows the results obtained for training the classifiers with the GTSRB [79] together with the parameters of each CNN. The accuracy results are obtained evaluating them on the test set.

From the Table 3.3 we can see that, all classifiers perform better when dropblock [178] is used instead of dropout [110]. If we compare the accuracies obtained using only dropout as regularization, the asymmetric kernels of the proposed Class\_CNN not only reduce the number of parameters but also increased the performance by 0.22% compared to the symmetric ones (Class\_CNN\_Symmetric), and by 0.33% with the CNN with residual blocks (Class\_CNN\_RB). On the other hand, comparing the accuracies obtained using



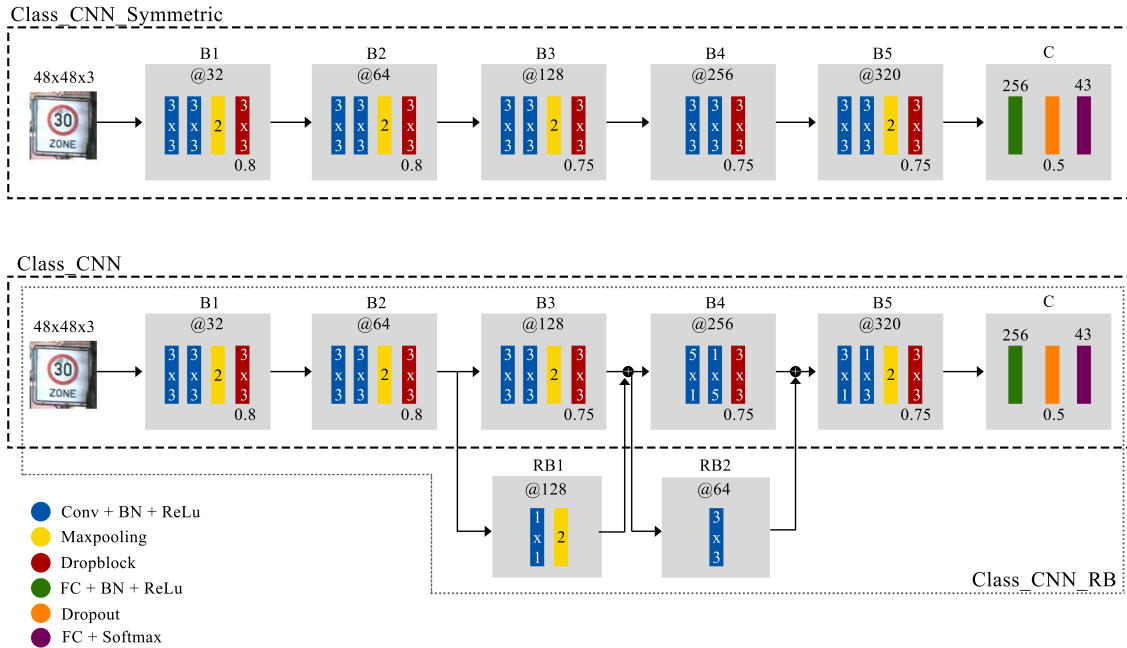


FIGURE 3.8 – Visual representation of Class\_CNN\_Symmetric, Class\_CNN and Class\_CNN\_RB.

TABLE 3.4 – Class\_CNN accuracy percentage results obtained on the GTSRB test set.

Trials	Accuracy (dropout)	Accuracy (dropblock)
1	99.41%	99.51%
2	99.33%	99.48%
3	99.37%	99.61%
4	99.30%	99.54%
5	98.83%	99.50%
Average	99.25% ± 0.21	99.53% ± 0.05

dropblock, the ones obtained from Class\_CNN\_Symmetric and Class\_CNN are almost the same. The possible reason behind this behavior could be due to the way dropblock discards the activations. Dropout discards randomly the percentage defined, while dropblock, turns off the activations with continuous similarities, adding more stabilization to the learning process. In both accuracy results, the Class\_CNN\_RB performs the worse, eliminating the possibility to use this strategy for not deep networks. The CNN with asymmetric kernels (Class\_CNN) proposed in this study, provides the best trade off between accuracy and complexity, proving its good performance.

Furthermore, in order to analyze the stability of the proposed Class\_CNN, we trained it 5 times with data-augmentation. We performed the same procedure using dropout and dropblock to compare how the two regularizations affect the model. Table 3.4 shows the results obtained for 50 epochs each called trials.

TABLE 3.5 – Accuracy percentage results obtained on the GTSRB and European test sets without and with performing data-augmentation.

Model	GTSRB		European	
	Original	Augmentation	Original	Augmentation
CNN_asymmetricK	97.88%	99.37%	98.48%	98.89%
Improved_CNN_8-layers	98.52%	99.37%	97.88%	98.99%
Class_CNN	98.51%	99.53%	98.25%	99.11%

Analyzing the results obtained in Table 3.4, we confirm that the regularization has an important role in the model stability. The learning variability in our proposed CNN architecture decreased from 0.21 to 0.05 using dropblock, reason why it was chosen. The best test accuracy obtained was 99.61% while the average was 99.53%.

### 3.5.2.2/ CNNs COMPARISONS

Now, after discussing the performance of our proposed Class\_CNN, it is time to compare it with the other two best CNN architectures which provide the best results in Table 3.2. We trained the three CNNs on both datasets and the results obtained are shown in Table 3.5. The testing accuracies obtained with the CNN\_asymmetricK model [181] are improved with data augmentation by 1.49% on the GTSRB dataset and 0.41% in the ETSD, while with the Improved\_CNN\_8-layers model, they are improved by 0.85% and 1.11% respectively. Regarding our proposed Class\_CNN, we obtained an improvement of 1.02% on the GTSRB test set and 0.86% on the ETSD test set.

The average human performance for detecting traffic signs on the GTSRB dataset is 98.84% as reported in [79]. The three CNNs trained in this study with data-augmentation surpassed the human performance with both datasets. For the URV model proposed by Aghdam et al. [158], we obtained 96.1% accuracy without data-augmentation while they reported 98.94% applying 12 transformations on the dataset. With this in mind, we can affirm that a classifier learns more robustly if the dataset comprises a wide variety of data situations.

### 3.5.3/ DISCUSSION

Due to the fact that the proposed ETSD comprises a wider range of situations and in consequence a larger number of training data, it also includes difficult classes to learn. Because of that, most of the accuracy results obtained for each model without data-



FIGURE 3.9 – Sign symbol classes grouped in Others category for the GTSRB and the ETSD.

TABLE 3.6 – Error percentage predictions by category on the GTSRB and ETSD test sets. Results are computed from the CNN models trained with data-augmentation.

Category	GTSRB				European			
	Signs in GT	CNN_asymmetricK	Improved_CNN_8-layers	Class_CNN	Signs in GT	CNN_asymmetricK	Improved_CNN_8-layers	Class_CNN
Danger	2790	1.25%	0.39%	0.79%	4626	1.75%	1.36%	1.23%
Priority	-	-	-	-	2946	0.17%	0.20%	0.24%
Prohibitory	5670	0.00%	0.83%	0.39%	8625	1.18%	0.94%	0.83%
Mandatory	1770	0.66%	0.83%	0.17%	2818	0.43%	0.64%	0.53%
Special Regulation	-	-	-	-	1550	1.35%	1.35%	1.03%
Information	-	-	-	-	59	0.00%	3.39%	0%
Direction	-	-	-	-	706	2.55%	3.12%	3.26%
Additional panels	-	-	-	-	392	0.77%	1.79%	1.02%
Others	2400	0.17%	0.11%	0.46%	208	0.96%	0.96%	0.96%
Total	12630	0.63%	0.63%	0.46%	21930	1.11%	1.01%	0.89%

augmentation (see Table 3.2) on the ETSD are almost the same or even lower than the ones obtained on the GTSRB. Nevertheless, with data-augmentation (Table 3.5), all models could improve their accuracies on both datasets but, on the ETSD, models barely achieved around 99%. We will analyze the predictions in both datasets with the three CNN models trained with data-augmentation to find out the reasons that made the classifiers failed.

Table 3.6 presents the relations of the incorrect classes by category per model and dataset. It is worth mentioning that, by naming convention, the 'Others' category does not represent the same classes on the GTSRB and the ETSD. Fig. 3.9 shows the classes covered in 'Others' category for both datasets. We present the errors by category because it is the common way in the literature [79, 83, 153, 167] to compare the differences between groups of classes with similar characteristics (shape, color). For the ETSD (Table 3.6), we can see that the CNN\_asymmetricK model has troubles learning the categories Danger, Prohibitory, Special Regulation and Direction with more than 1% error, while the Improved\_CNN\_8-layers model has troubles with Danger, Special Regulation, Information, Direction and Additional panels with more than 1% error as well. For our proposed Class\_CNN, we only have troubles in 4 categories : Danger, Special regulation Direction and Additional panels.

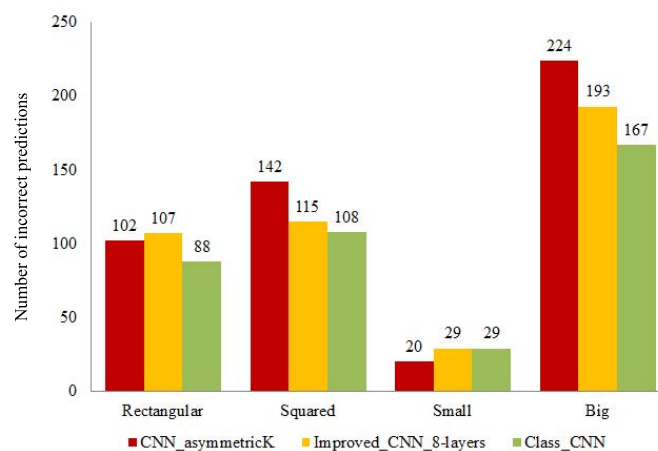


FIGURE 3.10 – Image size analysis for the incorrect predictions on the ETSD test set. Results are obtained with the CNN models trained with data-augmentation.

A deeper analysis for the incorrect predictions on the ETSD is performed to find out the characteristics of the traffic signs hard to recognize for the classifiers.

The first intuition for bad predictions was image size and aspect ratio. We counted the number of misclassified signs that were 1) rectangular or squared and 2) big or small. A squared sign is considered if its aspect ratio falls between 0.9 and 1.1, while small signs are considered if the image has less than 255 pixels. The latest parameter is set taking into account that the smallest image size on the GTSRB is 15×15 pixels (225 pixels).

As a reference, the total number of incorrect predictions with the CNN\_asymmetricK model is 244, for the Improved\_CNN 8-layers model is 222 while for our Class\_CNN is 196. Fig. 3.10 shows the relation of the incorrect predictions according to the parameters previously mentioned. There, we can see that only a few signs from the incorrect predictions are small for each of the classifiers (8.2% for CNN\_asymmetricK model, 13.06% for the Improved\_CNN 8-layers model and 14.79% for the Class\_CNN model), while almost half of the incorrect predicted signs are rectangular for the three classifiers. We considered these metrics because : 1) when the image size is small, even for humans, it is hard to distinguish the correct class ; and 2) when the image is rectangular, the classifier resizes it to a squared size suffering from information loss.

In the same way, we analyzed the predicted probabilities for the misclassified signs. We consider as uncertain predictions the ones which are incorrect and predicted with a probability equal or bigger than 0.9. The most uncertain predictions obtained with the CNN\_asymmetricK model were  $107/244 = 43.85\%$ , for Improved\_CNN 8-layers model were  $65/222 = 29.28\%$  while for the Class\_CNN were  $75/196 = 38.27\%$ . This kind of analy-

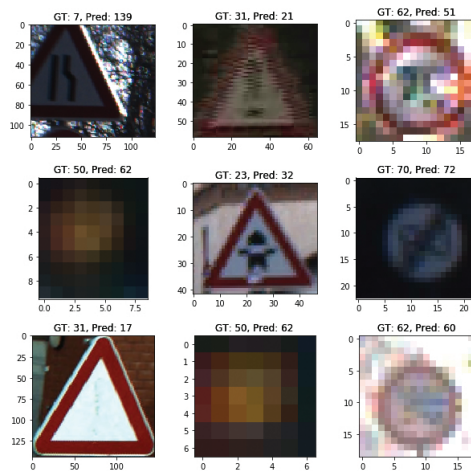


FIGURE 3.11 – Random sample of incorrect predictions of the ETSD with the CNN\_asymmetricK model trained with data-augmentation.

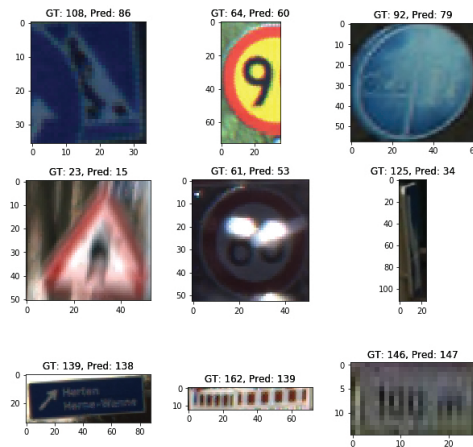


FIGURE 3.12 – Random sample of incorrect predictions of the ETSD with the Improved\_CNN\_8-layers model trained with data-augmentation.

sis can help a classifier refuse the prediction if the confidence probability is less than a certain threshold, however, in this approach, it is not applicable. As we are interested in the visual characteristics that make the signs difficult to classify correctly, we inspected all the incorrect predictions. Figures 3.11, 3.12 and 3.13 illustrate some of the incorrect predictions for the CNN\_asymmetricK model [181], Improved\_CNN\_8-layers model [159] and Class\_CNN model respectively.

After the visual inspection, we found that most of the misclassified signs possess the following characteristics :

- Strong motion blur.
- Incomplete signs (cropped).
- Occlusions.

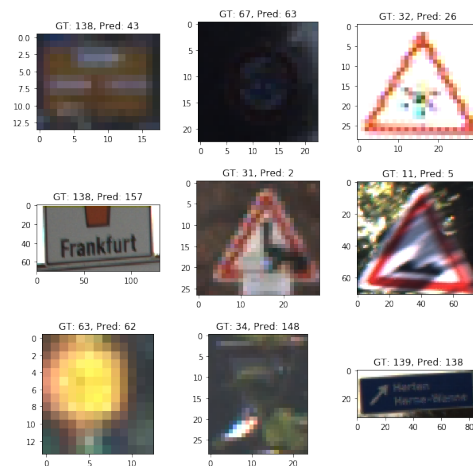


FIGURE 3.13 – Random sample of incorrect predictions of the ETSD with our proposed Class\_CNN model trained with data-augmentation.

- Strong shadows or highlights.
- Strong perspectives.
- Human added artifacts.
- Poor image quality.
- Aging.
- Very different aspect ratios (rectangular signs).

Most of the errors in Danger and Regulatory categories are due to the characteristics listed above. For the Informative category, the misinterpreted signs are mostly due to their visual complexity and to the very different aspect ratios. Informative signs contain text, which by nature, makes them the hardest ones to recognize. At the same time, their very different aspect ratios conduce to information loss once the classifier resizes them to a common input shape, normally, a squared shape. For example, in the Direction sub-category, most of the errors reside on confusion of class 139 (Direction to place) with class 138 (Advance directional signs) and vice-versa (see Fig. 3.12) due to the fact that both contain text and their appearances vary a lot.

Interestingly, no matter how many conditions the ETSD considers (see Fig. 2.10), there will always be hard situations for the classifiers to learn. In order to overcome this issue, image processing techniques can be used to enhance the visibility of an image and data-augmentation can be applied to improve the learning process generating more samples with different transformations.

In summary, our proposed CNN architecture is capable of classifying in our PC, in GPU mode, a traffic sign in 0.16 milliseconds which makes it optimal for the whole pipeline of

traffic sign recognition. Evaluating its performance, our Class\_CNN is capable of classifying correctly on the ETSD 98.48% traffic signs with a confidence probability bigger or equal to 0.9 and, 99.31% with the same conditions on the GTSRB. This parameter allows us to confirm that the architecture is robust and reliable enough to classify traffic signs.

### 3.6/ CONCLUSION AND FUTURE WORKS

In this chapter, we have reviewed, proposed and evaluated several CNNs architectures to perform traffic sign classification. Differently from the state of the art models, which focus only on symbol-based signs, we have considered traffic signs with text and with very different aspect ratios (ETSD). These two characteristics increase the complexity for a classifier to learn and distinguish appropriately the traffic sign classes. Despite this, such classes provide relevant information about the driving environment as : a) they can indicate a direction to take, b) alternatives routes when road works happen, c) provide complementary information to understand precisely the traffic sign (additional panels), d) limit the driving area with signs such as obstacle, access on the side, barrier, zone for pedestrians or bicycles, among others, or e) simple inform about certain situations like parking zones, residential areas, etc.

We have trained several state of the art CNN models with the GTSRB and the ETSD and, showed that accuracy drops when the classes complexity increases. At the same time, we demonstrated that Deep CNNs are not required to solve traffic sign classification. Instead, techniques like image preprocessing and data-augmentation are used to improve classification accuracy. The best accuracy results were obtained with our proposed Class\_CNN with 99.53% and 99.11% on the GTSRB and our ETSD test sets respectively.

However, all CNN architectures trained on the ETSD, exhibited the same behavior with classes based on text and with very different aspect ratios (most belonging to Informative category). They were the most challenging classes to learn due to the input definition of the CNNs architectures. Since CNN models require a fixed size (most of the time squared), information might be discarded when downscaling the image, or distorted from the original input. Simultaneously, the text signs were always confused with other classes due to their appearance variabilities. For example, directional signs and advanced directional signs were always confused between them and some additional panels as well.

In future work, we intent to take into account the class imbalance problem to improve recognition accuracy. In this regard we will : 1) take into account more transformations

for data-augmentation as performed in [158], 2) generate data, simulating night scenarios with different levels of brightness changes and noise additions 3) apply class independent transformations like horizontal flip to signs which allow changing one class to another and 4) vertical flip to signs that do not change meaning. Besides the classification, we will also attempt to 5) perform traffic sign detection in an image considering all traffic signs no matter their shapes, colors, and text, to provide a complete traffic sign recognition system for autonomous vehicles.





# VISUAL PERCEPTION SYSTEM FOR URBAN ENVIRONMENTS

## 4.1/ INTRODUCTION

Scene understanding is one of the main goals of autonomous driving systems. Extracting information from the environment and being able to process it correctly, involves several steps including data collection, data preparation, processing, and sometimes post-processing. The data collection and preparation was performed as introduced in Chapter 2 while part of the data processing for traffic sign recognition was carried out in Chapter 3 to continue with the work presented here.

A considerable amount of research has been dedicated to the development of computer vision tasks for safety enhancement [171]. Among them, rely the ones to locate and classify objects of different sizes such as road, lanes, traffic signs, traffic lights, vehicles and pedestrians.

Recently deep learning methods, in particular, Convolutional Neural Networks (CNNs) have shown very good performance on detection and classification tasks opening a door for researchers to explore solutions based on them. Nevertheless, the nature of the objects may lead to complicated cases, since, every object possesses different characteristics and appears in different sizes. For example, road, sky and buildings represent a considerable portion of the image making them easier to detect and recognize, while in the other hand, traffic signs, traffic lights and lane markings are harder cases due to the small size they occupied in the captured images. In addition, illumination changes, weather conditions and occlusions influence as well the learning procedure to succeed on accurate localization and in consequence classification.

Driver assistance systems are an example of applications which deal with the detection

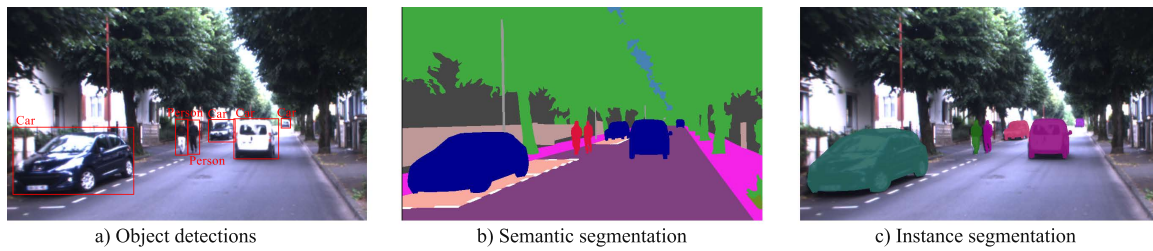


FIGURE 4.1 – Visual representation of the differences between object detection (objects enclosed with RoIs), semantic segmentation (dense pixel inference) and instance segmentation (dense pixel inference of the detected objects).

and recognition of small objects in the scene. Lane departure warning systems and traffic signs recognition are useful and important operations to increase safety. In fact, many tasks require the recognition of small significant objects, reason why it is of major interest to revise and evaluate methods which perform well for such tasks.

Detection approaches like R-CNN [99], Fast R-CNN [117], Faster R-CNN [126] are able to detect regions of interest (RoI) with their corresponding labels in a considerable amount of time. However, most of the time, these techniques require localization refinement for accurate results.

Other approaches like semantic segmentation, are used for environment perception to classify each pixel in the image with a corresponding label [161]. Their performances depend on the actual image size and, their accuracies on the size of the objects. They have proven to perform good on big objects while conducting poorly on small ones. When talking about detecting and recognizing objects in the scene, it is necessary to detect all objects that belong to a class, together with their individual instances, as some of them may require further classification, like traffic signs. Semantic segmentation approaches are able to classify each object of the same class, but fail to detect their different instances. In order to deal with the previously mentioned problem, instance segmentation came into play to not only detect individual objects in the scene, but also segment their masks (portion of the image pixels which actually belong to the object). Both approaches work differently. For instance, semantic segmentation classifies each pixel with a label, while, instance segmentation first detects objects in the scene as portions of the image and then, based on that, decide which pixels belong to the actual class. In this way, a single class possesses several instances which are useful for identifying or tracking single objects. Fig. 4.1 shows an example of the differences between object detection, semantic segmentation and instance segmentation.

Traffic signs are one of the classes represented with a single label for semantic segmen-

tation approaches, but require additional classification to identify the specific traffic sign class. For this reason, in this chapter, we propose a pipeline to perform environment perception for urban scenarios, recognizing objects in the scene into 15 categories using semantic segmentation, while at the same time, through instance segmentation, identification of certain classes like traffic signs, moving objects, traffic lights and crosswalks will be separated into their instances. The objective of this, resides in the goal to perform further processing. In our case, we will further identify the specific traffic sign classes with an additional proposed CNN.

We summarize the contributions to the following :

1. Proposed a whole pipeline to identify important traffic agents in urban environments.
2. Through fine-tuning of different semantic segmentation approaches, we analyzed and chose the one that better adapts to French urban scenarios, validating them on the proposed UTBM-2 dataset.
3. We trained Mask R-CNN [162], an instance segmentation approach, as the base detector on the Mapillary Vistas dataset [166] to identify 19 instance classes as possible objects of interest for autonomous driving systems. Later, with the weights obtained, we fine-tuned them on : the GTSDB, the proposed extended GTSDB and the proposed UTBM-2 dataset, to focus mainly on symbol and text traffic signs for German and French roads.
4. Evaluation of traffic sign detection and classification will be performed with a careful selection of 132 urban classes in the extended GTSDB and the UTBM-2 datasets.

The rest of this Chapter is organized as follows : Section 4.2 presents related works. Section 4.3 describes the proposed system for environment perception and traffic sign recognition. Section 4.4 continues with details about the datasets used, the training procedures and the analysis of the results obtained evaluating the system on French and German roads. Conclusions and future work are presented in Section 4.5.

## 4.2/ RELATED WORKS

We will discuss some related works regarding semantic segmentation for scene understanding. At the same time, works on traffic sign recognition will be reviewed, focusing particularly on detection as classification was discussed previously in Chapter 3.

#### 4.2.1/ SEMANTIC SEGMENTATION

Semantic segmentation is a popular problem in computer vision since, it is one of the high-level tasks that will allow to achieve complete scene understanding. It requires the estimation of a function  $F$  to map an input image  $I$  to an output label map  $L$ , with the same size as the input image. Each label index in the label map corresponds to a semantic class of the input pixel (e.g. road, sky, building, etc.). Because of its dense per-pixel inference and due to the success of CNNs in classification and detection approaches, semantic segmentation also adopts them to solve its complex problem.

The pioneering work to perform semantic segmentation was proposed by Long et al. [123] with the Fully Convolutional Network (FCN). Based on CNNs architectures, they replaced the fully connected layers with convolutional ones to output spatial maps instead of classification scores. The maps are up-sampled using fractionally stride convolutions called deconvolutions [65]. In this way, it is possible to infer a class label for each input pixel (dense per-pixel output). Their work is considered as a milestone because it proved that taking advantage of CNNs and their ability to learn feature representations automatically, the task could be solved in an end-to-end manner.

Apart from the FCN architecture which can convert any well known classification model - like AlexNet [73], VGG [109], GoogleNet [128] and ResNet [138] - into a model for dense segmentation tasks, other techniques were also proposed to solve this problem. Among them rely the ones that implement decoders [114] instead of using FCN, or the ones that integrate context information [170, 130, 175] for more accurate prediction.

SegNet [114] is one example of the methods that used decoders. In theory, it is composed of an encoder and a decoder. The first one refers to any CNN architecture removing the fully connected layers to produce low resolution image representations or feature maps. The decoder, on the other hand, needs to learn how to decode or map those low resolution images to pixel-wise predictions of the same size as the input image. In the case of SegNet, which is based on VGG16 [109], after removing the fully connected layers, 13 blocks of convolutional layers represent the encoder. Its decoder consists of a set of upsampling and convolutional layers. Each downscaled feature map is performed with max-pooling while the decoder has an upsampling layer for each of them. This in order to restore the original resolution and feed it to a softmax classifier to produce the final dense segmentation.

Although, the FCN and encoder-decoder approaches seem to perform good to solve the

problem, as soon as the image gets down-sampled with the CNNs, boundary information gets lost leading to similar object appearances and, in consequence, errors in predictions. For this reason, context information should also be considered when dealing with semantic segmentation.

Integrating context information can be done through different techniques. One of them is enlarging the receptive field-of-view of neural networks using dilated convolutions [130, 145, 170, 175]. Dilated convolutions, also known as *à-trous* (french word meaning with holes in) convolutions, allow expanding the field-of-view of kernel filters at any CNN layer without loosing resolution, which makes them a good strategy as no additional cost is encountered. Among the common CNNs that make use of this strategy are PSPNet [170] and DeepLab [96, 175] models.

Other works, have combined multi-scale features [148, 132] to include context information. Normally these approaches make use of multiple networks that target different scales and, at the end, merge the predictions to output a single result. Additionally, driven by the image pyramid, Chen et al. [175] proposed the Atrous Spatial Pyramid Pooling (ASPP) module to capture context information of different receptive fields. Zhao et al. [170] proposed the Pyramid Pooling Module (PPM) to fuse features under four different pyramid scales in the final layer feature map, for global scene prior construction. Besides that, fusing global and local feature maps [122, 146] extracted from different layers in the CNN architecture, also increases the segmentation accuracy.

Yu et al. [186] tried to deal with the semantic segmentation problem focusing on designing a CNN (BiSeNet) capable of dealing with the spatial and the context information through two different paths and, fusing the features with their proposed fusion module.

In order to deal with the boundary loss mentioned before, Conditional Random Fields (CRF) are used as post-processing step to refine the segmentation results [131, 175]. The CRF of DeepLab [175], models each pixel as a node in the field and employs one pairwise term for each pixel pair. In this way, the system is able to recover detailed structures of the object. In a similar manner, the work proposed in [115] uses a densely connected CRF to refine the results of the CNN applied to material classification.

In Section 4.3.1, we will analyze in detail some CNN models that make use of decoders, include contextual information and use CRF for boundary refinement to evaluate them later in Section 4.4 and choose the one that fits best for our system.

#### 4.2.2/ TRAFFIC SIGN RECOGNITION

Traffic sign recognition is usually carried out in two steps : detection and classification. The detection consists of localizing spatially in an image the region where the object of interest appears. Normally this region is represented through pixel coordinates called Region of Interest (RoI) and it can be detected through different approaches depending on the final aim. Classification, on the other hand, consists of predicting a label for the input, based on a series of features learned by the classifier. It is performed after the detection is carried out identifying the RoIs and using them as input to decide whether or not a RoI is a traffic sign and infer its specific class.

As traffic signs are grouped into categories with similar shapes and colors [1], a considerable amount of approaches have been proposed taking advantage of these key characteristics. Color-based detection methods [22, 34, 36, 59, 153, 151] segment the most used traffic sign colors (red, blue and yellow) converting RGB images to other color spaces in order to reduce illumination sensitivity and enhance the target hues to extract RoIs. For example, Maldonado et al. [36], Kuo et al. [34] and De la Escalera et al. [19] use the hue-saturation-intensity (HSI) color space to enhance red, blue and yellow colors using the hue and saturation components. Then color thresholds together with size and aspect ratios are used to segment traffic sign regions for further processing. Also, Yang et al. [153] exploit color for detection, but instead of HSI, they converted RGB values to Ohta space, proposing a color probability model to transform color images into probability maps to finally use Maximally Stable Extremal Regions (MSER) detector to keep only the most stable regions as RoIs.

On the other hand, as color is a key component but not suitable for outside environments due to illumination changes, shape-based approaches came into play discarding this information and focusing only on detecting certain contours like squares, circles or triangles to identify specific traffic sign categories. Edge detectors, Hough transforms [34, 48, 69, 59], radial symmetry voting [18, 25] as well as template matching [10] are the most common techniques. Edge based methods make use of Canny algorithms and Hough transforms to identify shapes. García Garrido et al. [69] detected Spanish circular and triangular signs with Canny edge detector and refined with Hough transforms from gray-scale images. Two Support Vector Machines (SVM) were used to further classify each shaped signs.

Radial symmetry voting is a variation of Hough transform that finds points of interest in an image and detects circular signs in an more efficient way. This kind of approach was

extended in [20] to detect also triangular, square and octagonal signs. Template matching as its name suggests, utilizes templates to scan the whole image and find similarities through distance transforms.

Some approaches use a combination of both, color and shape, to detect traffic signs [19, 34, 151]. In this case, color locates the signs roughly and the shape, filters out false positives that do not fall into the characteristics of certain categories. Additionally, other methods like support vector machines (SVM) [36, 153], clustering [103] or Artificial Neural Networks (ANNs) [94] are used in combination with color and shape to detect traffic signs. However, all the previously mentioned methods are limited to specific definitions and are sensitive to weather conditions, illumination changes, reflections, occlusions, rotations, etc., reasons that make them very ineffective for real world environments.

Besides shape and color methods, machine learning approaches have been used for traffic sign detection after the successful application in image classification and their adaptation to object detection. These methods treat detection as a classification task and their performance depends on the features selected. Hand-crafted features like saliency [129], local binary patterns (LBP) [107, 142], histogram of gradients (HoG) [92, 129, 153], integral channel features (ICF) [112] or aggregated channel features (ACF) [168] have been applied for this task. Nevertheless, it has been proven that automatically learned features through CNNs perform better than the hand-crafted ones. More comprehensive reviews about detection methods for traffic signs are presented in [77, 167].

Since 2013, CNNs have been used for object detection to first calculate some generic region proposals and then perform classification on the candidates. R-CNN [99] is one example of the formerly mentioned. It extracts regions through Selective Search [91] and with a CNN, feature maps are extracted for each proposal to be passed to bounding box regressors and SVM classifiers. However, due to the inefficiency of R-CNN, numerous efforts were made to improve this approach. Fast R-CNN [117] jointly optimizes classification and bounding box regression, replacing the SVM classifier with a Softmax layer. Faster R-CNN [126] improves the object proposal step introducing Region Proposal Networks (RPNs) which share full-image convolutional features making the detection suitable for real-time systems. Thanks to the object detection success of these approaches, researches have been inspired and different methods have been proposed for traffic sign detection. Quian et al. [147] focused on detecting traffic signs painted on the road using Maximally Stable Extremal Regions (MSER) [21] detectors and EdgeBoxes [113] (bounding box proposals using edges) to identify region proposals and pass them to Fast R-



CNN for further feature extraction, classification and bounding box regression. Huang et al. [179] made use of Generative Adversial Networks (GANs) together with Faster R-CNN to deal with the problem of detecting small objects like traffic signs. Their idea was to map features of the small traffic signs into large object features with the same feature distribution to be able to provide a more accurate prediction. Li and Wang [181] also used Faster R-CNN to detect candidate traffic signs and proposed a refinement of the candidate regions through color segmentation and Hough transforms to identify 4 shapes (circle, rectangle, triangle, octagon).

Until now and despite the good performance of detection approaches based on CNNs like Faster R-CNN, small size objects are not detected accurately and, in consequence, further processing is required. In order to deal with this issue, we use as a detection approach, Mask R-CNN [162]. Mask R-CNN is based on Faster R-CNN predicting the class, bounding box coordinates and a mask for each RoI. In our case, we aim to use the mask branch as a localization refinement for traffic signs. In this way, if the RoI predicted is not accurate enough as Li and Wang [181] encountered, the mask will discard the extra background pixels and a more accurate classification will be performed in further steps.

### 4.3/ VISUAL PERCEPTION SYSTEM

Our proposed system for environment perception is composed of two modules. The first one deals with the recognition and localization of 15 object classes in the scene through semantic segmentation. As small objects are not clearly recognizable and instances are not possible to obtain, the second module deals with these shortcomings using instance segmentation and focuses on traffic signs to identify the specific traffic sign classes. In other words, this last module performs traffic sign recognition with instance segmentation and a proposed CNN architecture. An overview of our proposed pipeline is illustrated in Fig. 4.2.

In the semantic segmentation module, we will provide a comparison study of four approaches described in subsection 4.3.1 for French urban environments. Regarding traffic sign recognition, we will compare our proposed system with the state of the art approaches dealing with detection and classification on the well known GTSDb [83] and the GTSRB [79] datasets. Additionally, and in order to deal with text traffic signs, we will provide comparison results between the GTSDb and the proposed extended GTSDb (see Section 2.4) to analyze the difficulties of handling both, text and symbol-based signs.

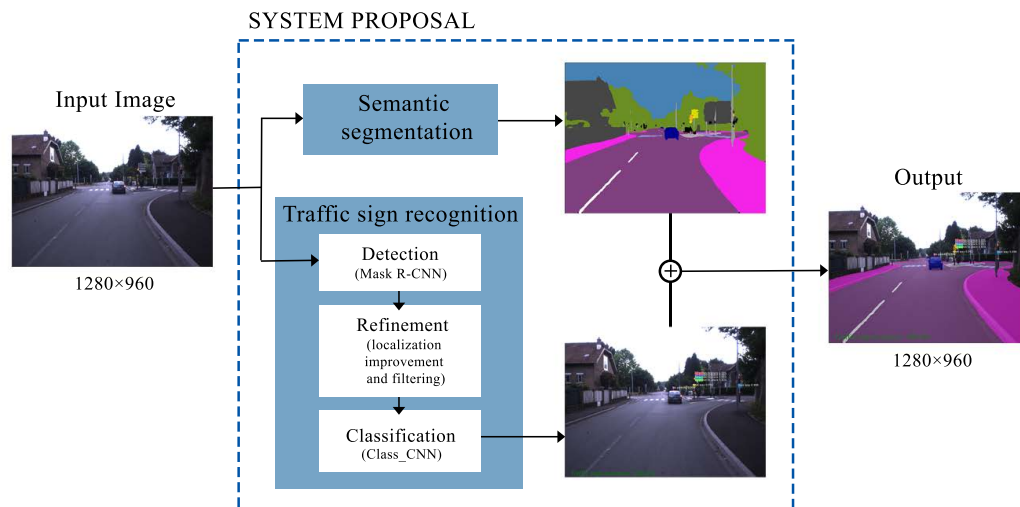


FIGURE 4.2 – System proposal pipeline composed of two modules : Semantic segmentation and Traffic sign recognition.

Lastly, evaluation of traffic sign recognition will be performed on French environments with the UTBM-2 dataset.

As a result, the output of both modules provides an accurate scene interpretation of urban environments.

#### 4.3.1/ ENVIRONMENT PERCEPTION THROUGH SEMANTIC SEGMENTATION APPROACHES

One of the primary motivations for semantic segmentation is road scene understanding. It requires the ability to model the appearance, shape and understand the spatial-relationship (context) between different classes like road and sidewalk. At the same time, it should be able to delineate object boundaries independently of their sizes. Several methods [161] have been proposed based on CNNs to deal with all these issues.

In this section, we will describe four semantic segmentation models that use different techniques to perform dense per-pixel inference. These are : SegNet [114] which uses an encoder-decoder method, PSPNet [170] which fuses multi-scale features, DeepLab [175] that fuses multi-scale features + boundary refinement and, BiSeNet [186] which deals with spatial and context information separately to fuse the features.

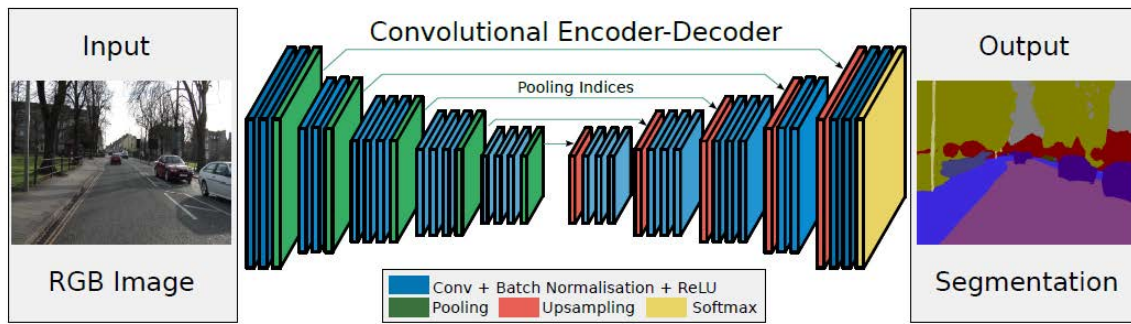


FIGURE 4.3 – SegNet architecture extracted from [114].

#### 4.3.1.1/ SEGNET

Bandrinarayanan et al. [114] proposed SegNet, a semantic segmentation network. Its structure is based on an encoder-decoder network. The encoder is topologically identical to VGG16 [109] removing its fully connected layers. The key component of SegNet relies in the decoder network which consists of a series of decoders, each corresponding to one encoder. In other words, the decoders used the max-pooling indices of the corresponding encoder to perform non-linear up-sampling of the feature map. Fig. 4.3 illustrates the SegNet architecture.

The encoder network consists of 13 convolutional layers after removing the fully connected ones from VGG16. Consequently, the decoder is also composed of 13 deconvolutional layers, for which at the end, the output is fed to a multi-class softmax classifier to produce class probabilities for each pixel.

Each encoder consists of a convolutional layer to produce feature maps. Then, these feature maps are batch normalized [120] and activated with ReLU [54]. After that each map is downscaled using max-pooling and saving only the indices (locations of the maximum feature value in each pooling window). For the decoders, each feature map is upsampled using the memorized indices to produce sparse maps. After this, each map is convolved with a trainable decoder filter bank producing dense maps. Next, the dense feature maps are followed by batch normalization and ReLU activations as performed with the encoders. At the end, the output of the last decoder is passed to a softmax classifier to predict  $K$  (number of classes) output probabilities for each pixel value. The class with maximum probability will define the final class.

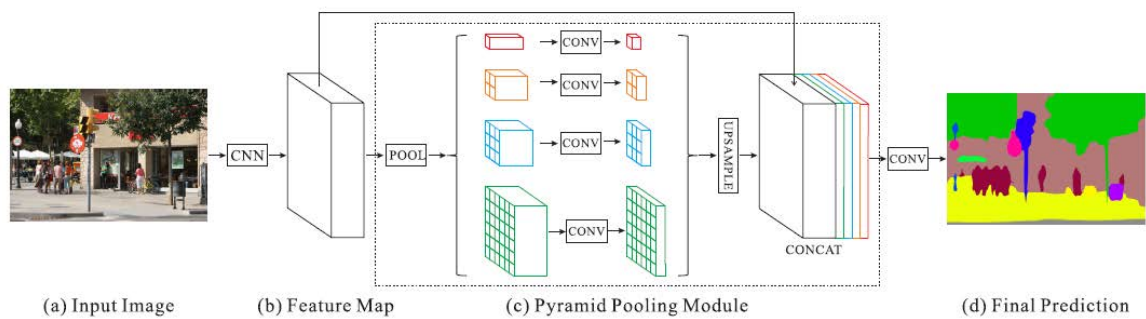


FIGURE 4.4 – Overview of the PSPNet architecture [170]. The feature map in (b) corresponds to the map obtained after removing the fully connected layers of ResNet [138].

#### 4.3.1.2/ PYRAMID SCENE PARSING NETWORK (PSPNET)

Zhao et al. [170] decided to make use of the pyramid structure to capture contextual information and improve the segmentation results. The architecture of their proposed PSPNet is illustrated in Fig. 4.4.

PSPNet makes use of dilated convolutions FCN [123] to keep dense feature maps resolution. It is based on ResNet [138] model to extract the final feature map. This map is  $1/8$  of the input image size. The pyramid pooling module takes this map as input and produces four feature maps performing pooling with different sizes ( $1 \times 1$ ,  $2 \times 2$ ,  $3 \times 3$  and  $6 \times 6$ ). Each of these pooled feature maps is then convolved with a kernel size of  $1 \times 1$  to reduce dimensionality. Once the four feature maps are convolved, they are up-sampled using bilinear interpolation to recover the same size as the original input feature map obtained from ResNet.

Additionally, and in order to include global contextual information, the feature map obtained from ResNet and the four feature maps computed with the pyramid pooling module are concatenated and convolved one more time to generate the final prediction map. As mentioned before, the map is  $1/8$  of the image input size, for which the convolved final feature map is up-sampled by a factor of 8 and passed to the final softmax classifier to output the probability map with the  $N$  number of classes for each input pixel.

#### 4.3.1.3/ DEEPLAB

Chen et al. [175] proposed a deep convolutional neural network (DCNN) architecture to perform semantic segmentation incorporating context information with their proposed atrous spatial pyramid pooling (ASPP) to robustly segment objects at multiple scales.

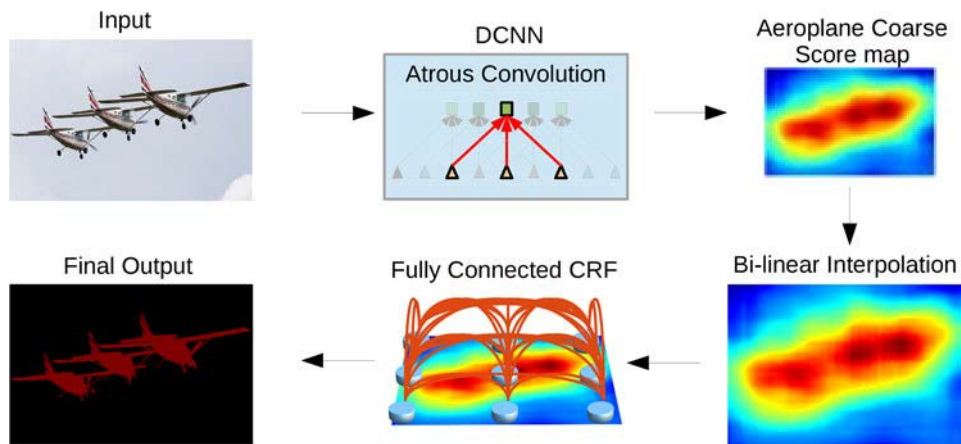


FIGURE 4.5 – DeepLab architecture [175].

Moreover, they refined the object boundaries using a fully connected Conditional Random Field (CRF). Their model architecture is illustrated in Fig. 4.5.

The backbone of their model is either VGG-16 [109] or ResNet-101 [138]. In our case, we will use ResNet-101 for training. The fully connected layers of the backbone architecture are transformed to fully convolutional layers (FCN). At the same time, through the atrous convolutions, the final feature map is down-sampled from 32x to 8x without any additional cost. The dilated convolutions are transformed from a kernel size  $k \times k$  to a kernel size  $k_e = k + (k - 1)(r - 1)$ , with rate  $r$  which introduces  $r - 1$  zeros between consecutive filter values.

Furthermore, contemplating different scales of feature representations, have proved to improve segmentation results due to the inclusion of context information. Through their proposed ASPP technique, inspired by the Spatial Pyramid Pooling in [101], multiple features are extracted and further processed in separate branches to generate the final feature map. Once obtained, this map is up-sampled by a factor of 8 to recover a feature map with the same original image size.

Due to the convolution and max-polling operations, the final feature map possesses a lot of smoothness which makes it hard to delineate object boundaries in the final prediction. To overcome that limitation, Chen et al. integrated in their architecture the fully connected CRF model of [60] and showed that it is remarkably successful for accurate semantic segmentation results.

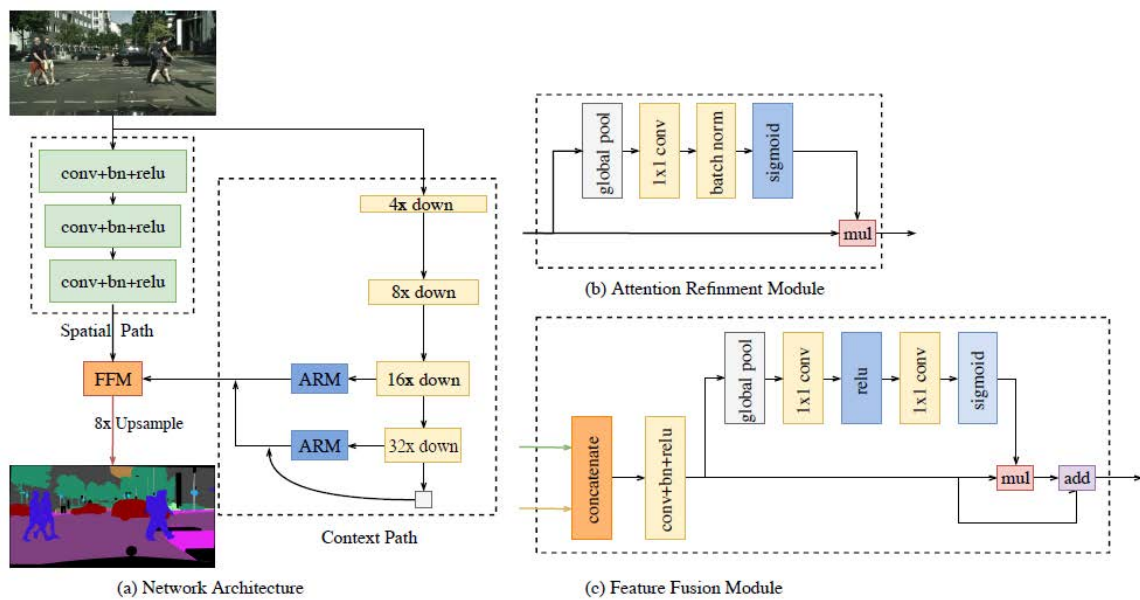


FIGURE 4.6 – Overview of the Bilateral Segmentation Network (BiSeNet) extracted from [186].

#### 4.3.1.4/ BILATERAL SEGMENTATION NETWORK (BISENET)

Yu et al. [186] focused on addressing the semantic segmentation problem dividing their CNN architecture into two paths. One is able to deal with the spatial information and to generate a high resolution feature, while the another one, is capable of handling the context information to, at the end, fuse both feature maps. Their architecture is presented in Fig. 4.6.

The spatial path (SP) contains three major layers, each corresponding to a convolution with stride 2 + batch normalization [120] + ReLu activations [54], extracting a feature map of 1/8 of the original image size. This map encodes rich spatial information of the input.

The context path (CP) models the context information using Xception [160] as backbone to down-sample fast the feature map and obtain large receptive fields. Three feature maps are obtained from the Xception as seen in Fig. 4.6a. The authors proposed as well an attention refinement module (ARM) to refine the features through global average pooling.

Lastly, the features obtained from both paths are fused though the feature fusion module (FFM) as shown in Fig. 4.6c. The authors claimed that performing global pooling after the concatenated convolved feature map, the weight vector obtained is able to re-weight the features and provide a correct feature representation of the input image. The final feature map is upsampled and passed to a softmax to provide the segmented output image.

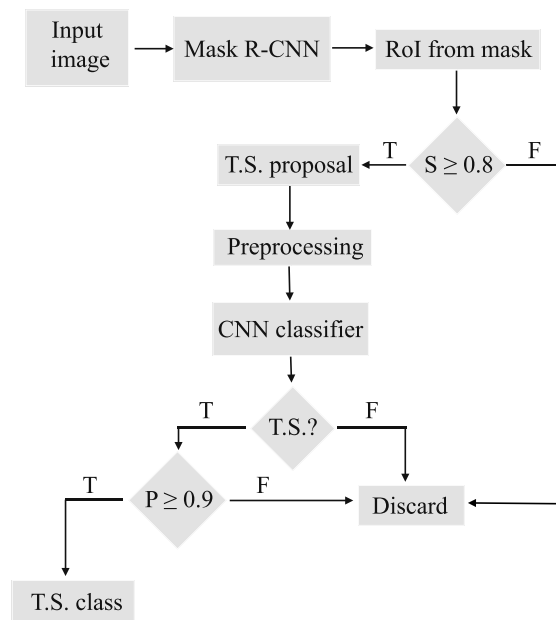


FIGURE 4.7 – Pipeline of our proposed traffic sign recognition module. T.S. refers to Traffic Sign. The Mask R-CNN block represents the detection module while the refinement is composed of the RoI extraction and class score ( $S$ ) filtering. The CNN classifier block makes reference to the classification module that will output a traffic sign class only if the predicted probability ( $P$ ) is at least 0.9.

All the previously described networks will be trained and evaluated in Section 4.4 to perform environment recognition in French urban scenarios with the proposed UTBM-2 dataset.

#### 4.3.2/ TRAFFIC SIGN RECOGNITION

Our traffic sign recognition module is illustrated in Fig. 4.7. The detection is capable of localizing the RoI where the sign is present. In our case this step is carried out by Mask R-CNN [162]. The advantage of using this approach resides in the localization refinement of the traffic sign performed at the same time when the mask is obtained. The classification, on the other hand, is accomplished with the proposed CNN to categorize the traffic signs and provide the specific classes for each category. This step takes as input the filtered output of the detection approach, pre-processes the image and labels each object with a corresponding class.

In the following sub-subsections, we describe in details the detection, refinement and classification tasks.

## 4.3.2.1/ TRAFFIC SIGN DETECTION

Traffic signs in real scenes have large differences in color, shape and size, reasons that make them hard to detect. Fortunately, deep learning approaches came into play with CNNs extracting automatically complex features without the need to design or chose manually the ones that will perform better to detect traffic signs.

We chose Mask R-CNN [162] as a general detector to identify static and moving objects seen in outside environments, but in this work, we only focus on traffic signs. This approach is based on Faster R-CNN [126] which detects objects in two stages : 1) propose candidate regions (RoI) through a Region Proposal Network (RPN) and 2) extract features for each RoI using the RoIPool layer to perform classification and bounding box regression. On the other hand, Mask R-CNN introduces a new branch to segment a binary mask for each RoI and replaces the RoIPool layer with a RoIAlign layer. The whole pipeline is illustrated in Fig. 4.8.

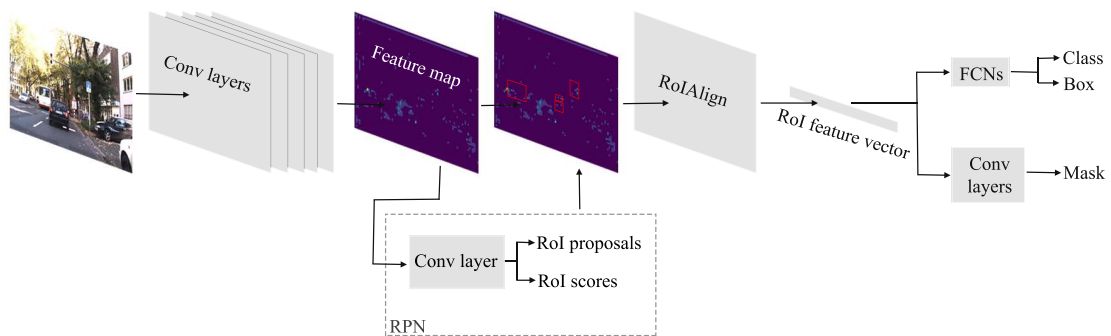


FIGURE 4.8 – Mask R-CNN data flow.

The input image is passed to a series of convolutional layers (based Conv layers) that output a feature map. In our case, ResNet-101 architecture was used together with Feature Pyramid Network (FPN) as feature extractor. Then, region proposals are obtained with the RPN. This network uses a  $n \times n$  spatial window to slide it through the whole feature map. Each window is mapped to a lower dimensional feature which serves as input to two fully connected layers : a box-regression layer and a box-classification layer.  $k$  region proposals, called anchors, are predicted at each sliding window location where the box-regression layer outputs  $4k$  values (box coordinates) and the class-regression layer  $2k$  scores (foreground or background). Normally, the anchors predicted are centered on the window and set with different sizes and aspect ratios (normally 3 each). In order to reduce redundancy of the overlapping region proposals, non-maximum suppression (NMS) is used based on the RoI scores. Only the regions with at least 0.7 IoU are then forwarded



to the detection and mask branches.

An intermediate layer to extract the features for each RoI from the image feature map is carried out by RoIAlign. This layer replaced RoIPool to overcome the feature misalignment introduced by quantization while mapping the extracted feature map into fixed size. Instead of quantizing the smaller feature map using max pool into a certain number of bins (e.g.  $7 \times 7$ ), RoIAlign samples each bin into 4 locations and computes their value by bi-linear interpolation from the nearby grid points. In this way, each bin aggregates the result of the 4 points using maximum or average operation to extract the exact values of the input features. This substitution leads to a more precise per-pixel segmentation and box regression [162].

Following the data flow in Fig. 4.8, each RoI feature vector is fed to the detection and mask branches to obtain its corresponding bounding box coordinates  $(x,y,w,h)$ , its score class and its respective binary mask, where  $(x,y)$  indicate the top-left corner and  $(w,h)$  the width and height of the box.

The decision of choosing this detector resides on the following reasons :

- It is based on Faster R-CNN [126], a fast detector, which has been used in several works [147, 134, 165, 179, 181] to identify traffic signs due to its capacity to detect small objects. In [173], the authors compared several architectures to detect traffic signs, specifically German signs from the GTSDB [83], and proved that, among the detectors (not including Mask R-CNN), Faster R-CNN outperforms the others no matter the backbone CNN used as feature extractor.
- Detection evaluation in [162], showed that the replacement of RoIPool (Faster R-CNN) with RoIAlign layer (Mask R-CNN), improved significantly the results.
- No further processing of the Rols is required to detect certain shapes [181] since the mask branch serves as localization refinement.
- The detector is not constrained to certain shapes like all the previous works mentioned in Subsection 4.2.2. Any traffic sign shape (circle, rectangle, triangle, square, octagon, etc.) will be detected and further classified.
- Its implementation is publicly available for different deep learning frameworks making it easy to use, adaptable to specific tasks and reproducible.

In this work we used the implementation made for Tensorflow [156] and trained the detector on the Mapillary dataset [166]. Details about the training and evaluations are presented in Section 4.4.

#### 4.3.2.2/ TRAFFIC SIGN REFINEMENT AND CLASSIFICATION

Once the outputs of Mask R-CNN are obtained, the refinement module extracts the exact RoIs using mask outputs and only keeps the regions for which class scores of being a traffic sign are bigger or equal than 0.8. This filtering is done to reduce the number of mistaken objects as traffic signs and to pass to the classifier a cleaner set. It is worth mentioning that the class scores do not influence the detection results but they are used for filtering because they provide a good hint to discriminate objects which do not belong to the traffic sign class.

The goal of the classifier is to identify properly the traffic signs classes and discard the false positives (background RoIs) for the recognition system. Traditionally, this filtering is done in the detection part, but as detectors are never perfect, the classifier needs to account for this as well. Fortunately, with the capability of CNNs, this task can also be done adding to the classifier an extra class for learning background objects. Here we will make reference to our proposed Class\_CNN described in Chapter 3, and we will introduce another CNN capable of learning traffic sign categories (Cat\_CNN). The performance of our deeper CNN (Class\_CNN) is competitive with the state of the art CNNs for traffic sign classification. Any of the CNNs is referenced as the classification module (CNN classifier) in Fig. 4.7.

As a reminder, our proposed Class\_CNN is based on 5 blocks to identify the specific class and from there, we extracted another CNN with 3 blocks to identify the traffic sign category (Cat\_CNN). The category classifier (Cat\_CNN) shares the same definition of blocks 1 and 2 from the class classifier (Class\_CNN) changing the third block with a convolutional layer of size 80. Fig. 4.9 illustrates both CNN definitions previously mentioned and Table 4.1 and Table 3.1 (Chapter 3) show specific details.

In Table 4.1, the output is set to 5 in order to recognize the 4 categories defined in the GTSDDB [83] (Danger, Prohibitory, Mandatory and Others) plus a background class. The category classifier has fewer layers compared to the class classifier, because recognizing the category is simpler than the class, as mostly shapes and colors are necessary. It is well known that the first few layers of a CNN detect lines and corners, while deeper layers learn more complex features. For this reason, only three modules are used for the category classifier whose details are shown in Table 4.1.

For comparison purposes and in order to evaluate the traffic sign recognition module, we trained the class classifier (Table 3.1 in Chapter 3) with 43 output classes according to

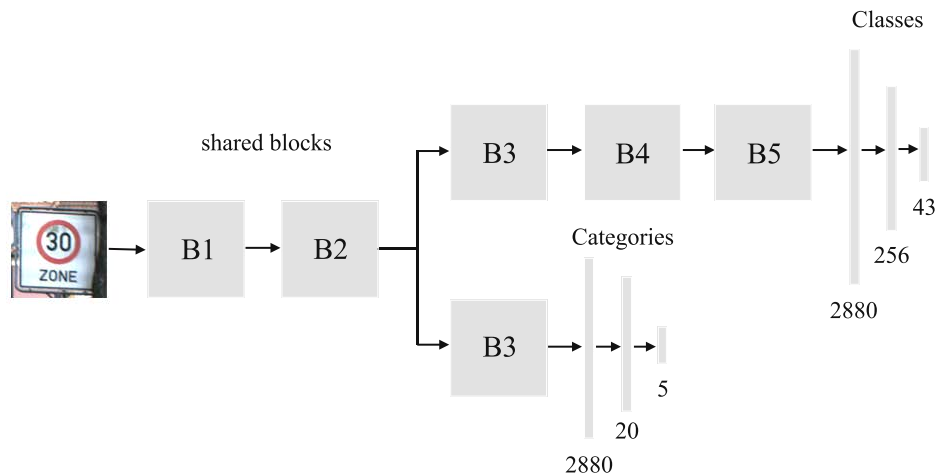


FIGURE 4.9 – CNN structure represented by blocks for the category and class classifiers. Block 1 and 2 (B1 & B2) are the same for both classifiers.

TABLE 4.1 – Architecture of our CNN for traffic sign categories identification (Cat.CNN). 'chan' makes reference to the number of channels, 'kS' to kernel size, 'std' to stride and 'kpProb' to the probability of keeping the original feature.

	type	outMap	chan	kS	std	keep prob
I	input image	48 × 48	3	-	-	-
B1	convolution	48 × 48	32	3 × 3	1	-
	convolution	48 × 48	32	3 × 3	1	-
	maxpooling	24 × 24	32	2 × 2	2	-
	dropblock	24 × 24	32	3 × 3	-	0.8
B2	convolution	24 × 24	64	3 × 3	1	-
	convolution	24 × 24	64	3 × 3	1	-
	maxpooling	12 × 12	64	2 × 2	2	-
	dropblock	12 × 12	64	3 × 3	-	0.8
B3	convolution	12 × 12	80	3 × 3	1	-
	maxpooling	6 × 6	80	2 × 2	2	-
	dropblock	6 × 6	80	3 × 3	-	0.75
C	fully connected	1 × 1	20	-	-	-
	dropout	1 × 1	20	-	-	0.5
	fully connected	1 × 1	5	-	-	-

the GTSRB plus a background class. Furthermore, we trained four more CNNs with the same architecture : one for each category of the GRSDB (Danger, Prohibitory, Mandatory and Others) and another CNN with a chosen subset of urban/rural classes extracted from our proposed ETSD [184] described in Chapter 2.2. In the same way, we trained one CNN with the category classifier architecture (Table 4.1) with 5 outputs to recognize the GTSDB categories plus background. The experimental results are provided in Section 4.4.

## 4.4/ EXPERIMENTAL RESULTS

In this section, we will evaluate both modules on French and German roads. The evaluation for the environment perception will be performed only on French urban scenarios, while for the traffic sign recognition module, experiments will be performed on French and German roads.



We will start with a brief description of the datasets used for evaluation to continue with details about the training procedures for each module. We will finish with the actual performances obtained between the different semantic segmentation methods evaluated on UTBM-2 test set and, a comparative study between our traffic sign recognition results and the ones reported by the competitors.

### 4.4.1/ DATASETS

#### 4.4.1.1/ SEMANTIC SEGMENTATION DATASET

The evaluation of the environment perception module performed through semantic segmentation considers only French urban scenes. Our proposed UTBM-2 dataset described in Chapter 2.3 consists of 541 images of resolution  $1280 \times 960$ , labeled with 27 classes.

For our experiments, we considered only 15 object classes :

- |   |  |  |
|---|--|--|
| 1. building    | 6. lane marking   | 11. refugee island      |
| 2. pole        | 7. parking block  | 12. vehicle             |
| 3. sky         | 8. sidewalk       | 13. bicycle_motorcycle  |
| 4. vegetation  | 9. crosswalks     | 14. traffic lights      |
| 5. road        | 10. humans        | 15. traffic signs       |

We fused building class with 2 others (fence and bridge), vehicle class with 3 (car, bus, trailer), human class with pedestrian and bicyclist, traffic signs with symbol and text signs classes and vegetation together with tree class. The rest of the classes not mentioned previously and included in the UTBM-2 labels, are fused as the extra void class. The color representations used for each class are taken from the Mapillary [166] definition.

We have decided to keep some classes as the originals due to the importance they provide for the environment perception. For example, road, sidewalks, parking blocks and

refugee island are relevant to distinguish for an autonomous vehicle in order to know the drivable territory. In the same way, lane markings intend to guide the vehicle to keep in its corresponding lane, while traffic signs and traffic lights tend to provide traffic rules while driving. Crosswalks were added to the classes of interest to distinguish the most dangerous zones where pedestrians tend to appear while crossing the road. All these classes will be evaluated in Section 4.4.3 considering only the 109 images of the UTBM-2 test set.

#### 4.4.1.2/ TRAFFIC SIGN DETECTION AND CLASSIFICATION DATASETS

In order to evaluate the proposed traffic sign recognition module, we use the GTSRB [79] and the GTSDb [83] to compare our method with other approaches. Nevertheless, these benchmarks are not representative enough for real world applications since 1) they include only pictographic/symbol based traffic signs with regular shapes and colors, 2) important classes are discarded like temporal signs, text-based signs, supplemental panels, etc., 3) symbol variations are not contemplated neither inside the same country nor for multi-country purposes (intra-class variability), and 4) images used for detection present considerable big traffic signs.

Hence, our proposed datasets, the extended GTSDb (see Chapter 2 Section 2.4) and UTBM-2 (see Chapter 2 Section 2.3), will be used as well to handle the shortcomings mentioned before when evaluating traffic sign recognition.

Furthermore, we extracted a urban subset of traffic signs from the ETSD [184] to handle intra-class variability (shortcoming three), symbol and text based signs, and multi-shape and multi-color signs for classification purposes. We carefully selected a group of 132 classes representative of urban/rural environments. The selection was made after a rigorous examination of frequent signs appearing in the GTSDb and UTBM-2 datasets. Fig. 4.10 shows the classes contemplated in this study which are analyzed further in Section 4.4.4.

In summary the datasets that will be used for traffic sign detection (Mask R-CNN) are : the GTSDb, the extended GTSDb and UTBM-2. Where the first one includes only symbol signs (4 categories) and the last two include more complex signs (9 categories including text traffic signs). Regarding traffic sign classification, we will use the GTSRB and the urban set from the ETSD.



(a) Danger warning signs



(b) Regulatory - Priority



(c) Regulatory - Prohibitory



(d) Regulatory - Mandatory



(e) Regulatory - Special regulation



(f) Informative - Direction



(g) Informative - Additional panels



(h) Others

FIGURE 4.10 – Urban/rural subset selection of the ETSD. (f) and (g) correspond to text-based signs while the rest make reference to symbol-based signs.

We emphasize the need to include text traffic signs (directional and additional panels si-

gns) or signs with very different aspect ratios (others category) since they provide useful information for the autonomous vehicle to interpret correctly traffic scenarios. The limitations and challenges encountered with these type of signs were discussed in Chapter 3. We will see how the performance drops when the detection and classification on the GTSDb dataset include other categories than symbol-based signs. We will compare the results obtained for the detection and classification on the GTSDb dataset, its extended version and UTBM-2.

#### 4.4.2/ TRAINING

All models are trained in GPU mode using a NVIDIA GeForce GTX1080Ti with 11GB of memory, an IntelCore i7K-8700K (6 cores 12 threads, 12 Mb cache memory) processor and RAM of 32GB.

##### 4.4.2.1/ SEMANTIC SEGMENTATION APPROACHES

For training the four semantic segmentation models described in Section 4.3.1, we made use of the semantic segmentation suite in Tensorflow<sup>1</sup> which contains the implementation for the four models.

We set the input image size to the original of the dataset,  $1280 \times 960$  pixels, to segment 15 classes. In order to increase the number of images for the training set, we perform data augmentation flipping the image horizontally and applying  $\pm 10$  degrees of random rotation. The mean image was subtracted from the dataset to account for normalization.

All the models were configured to use ResNet101 [138] as the backbone CNN together with the pre-trained weights in ImageNet [127]. The number of epochs to train each model was fixed to 100 with a batch size of 1. This small batch size was chosen according to the capabilities of our graphic card. To train all the variants we use the Root Mean Square Propagation (RMSprop) optimizer [81] with a fixed learning rate of 0.0001 and momentum of 0.995. A validation set of 54 images is used to ensure the learning and prevent over fitting.

The best weights for each model are chosen according to the validation loss obtained at every epoch. When this loss starts increasing, it means the model is over fitting and it is time to stop the training. In consequence, the weights obtained before this behavior

---

1. <https://github.com/GeorgeSeif/Semantic-Segmentation-Suite>

occurred, will be the ones that reached the global minimum and the picked ones for testing.

#### 4.4.2.2/ TRAFFIC SIGN DETECTION

We used the implementation of Mask R-CNN in Keras and Tensorflow<sup>2</sup>. This implementation uses ResNet101 with Feature Pyramid Networks (FPN) as backbone for feature extraction. The ResNet architecture divides its convolutional layers into 5 stages which are used later by the FPN to generate the final feature map. This feature map is composed of 5 pyramid stages which are convolved with 256 filters using a kernel size of  $3 \times 3$ . Once the feature map is extracted, the Region Proposal Network (RPN) takes it as input and feeds it into a convolutional layer of 512 filters with kernel size of 3 to generate 9 region proposals (using 3 scales and 3 aspect ratios [162]) for every window size location in the feature map. As the feature map is composed of 5 stages (ResNet and FPN), every stage is passed to the RPN network to extract Rols in different scales together with their probabilities (Rol scores) of being foreground or background. Considering that the positive Rols may be overlapping, non-maximum suppression is used to obtain only one.

We set to 255 the number of maximum detected Rols per image, where a positive output is selected if there is at least 70% overlap with the ground truth and discarded it if the overlap is less than 30%. Subsequently, the target Rols are passed to the RoIAligning layer (see Fig. 4.8) to extract fixed size features per proposal and feed them into the class and mask branches. For the classification branch, the Rol feature is passed to 2 convolutional layers with 1024 filters of size  $3 \times 3$  and  $1 \times 1$  followed by BN [120] and ReLU activations [54]. This convolutional layers are shared for the class and box heads. The class head uses Softmax activation to output the probability of the Rol to belong to a certain class (traffic signs), while the bounding box head utilizes linear activation to obtain the spatial locations of the Rol. The mask branch, on the other hand, takes the Rol feature and performs 4 convolutional operations with 256 filters using kernel size  $3 \times 3$ . In the same way as in the classification branch, each convolutional operation is followed by BN and ReLU activations. The mask for every Rol is obtained in the last layer with a per-pixel Sigmoid activation in order to have a binary mask.

Taking into account that the GTSDDB contains only 600 images for training, we first trained Mask R-CNN with the Mapillary Vistas dataset [166] to obtain some initial weights and fine-tune them later. This dataset contains 25,000 high resolution images representing

---

2. [https://github.com/matterport/Mask\\_RCNN](https://github.com/matterport/Mask_RCNN)



street-level scenarios. It is labeled in a per-pixel manner with 64 semantic classes and 37 instance ones. We chose this dataset for training as 1) it contains only driving scenarios, 2) traffic signs class contains symbol and text-based signs, 3) a diversity of scenarios is provided (urban, rural, highways), 4) the high image resolution allows to choose the proper size according to the needs of the task and 5) the number of images is enough to train a robust detector/classifier.

The Mapillary dataset is divided into 18,000 / 2,000 images for training and validation respectively. For our approach, we set the training size to  $1280 \times 960$  discarding all images in the dataset that do not fit our requirement. 17,979 images were used for training Mask R-CNN, while 1995 were used for validation. We chose carefully 19 instance classes comprising static and moving objects of interest to detect objects like cars, pedestrians, traffic lights, pedestrian crossing, traffic signs among others.

The training procedure was set to 32 epochs using Stochastic Gradient Descent (SGD) as the optimizer with learning rate equals to 0.01, momentum of 0.9 and weight decay of 0.0001. Due to the image size and capabilities of our GPU, only 1 image per batch was used. The ImageNet weights, trained in ResNet50, were used as initialization for fine-tuning all layers of Mask R-CNN with the Mapillary dataset. This procedure took 18 days, 23 hours, 29 minutes and 49 seconds.

Once the Mapillary weights were obtained, we continue to the second training stage to fine-tune the previously obtained weights with the original GTSDb [83], the proposed extended GTSDb and the proposed UTBM-2 datasets. The training parameters were almost the same changing the number of epochs to 50 for the three datasets. The training and validation steps for the GTSDb were set to 600 and 300 respectively, while for the UTBM-2 dataset, they were set to 389 and 42 (number of images in the train and validation sets). These training procedures took approximately 7 hrs each for the GTSDb and its extended version, whereas for the UTBM-2 dataset, it took around 5 hrs. The minimum validation loss for each dataset was obtained at epoch 46 for the original GTSDb, at epoch 45 for its proposed extended version and, at epoch 20 for UTBM-2.

The learning time was quite long due to the number of parameters that Mask R-CNN has (63,830,282), the image size ( $1280 \times 960$ ) and the batch size (1) used for training.

#### 4.4.2.3/ TRAFFIC SIGN CLASSIFICATION

We trained 7 classifiers with the two CNN architectures described in Section 4.3.2.2 using Keras and Tensorflow back-end. The training procedure is almost the same as we performed for the Class\_CNN in Chapter 3; the only difference is that we have included a background class to discard some false matches and we have trained the classifiers for 100 epochs instead of 50 with data augmentation. The preprocessing and training parameters are left unchanged. It is important to mention that the type of preprocessing influences the performance of the classifier. For instance, we have trained Class\_CNN applying histogram equalization in the lighting channel, adaptive histogram equalization and without any preprocessing, obtaining the best accuracy results with histogram equalization.

The background class was generated using the negative anchors from the Mask R-CNN on UTBM-2 and GTSDb. We have saved 20 Rols per image as background class and inspected them to discard the non appropriate ones (covering almost the whole image or Rols including traffic signs).

We have trained Class\_CNN on the GTSRB (43 + background classes) with data augmentation. In the same manner, we trained the urban/rural subset (132 + background class) subtracted from the ETSD [184] and used it to evaluate the detection and classification performances. The results obtained are presented in the Section 4.4.4.

In the literature, there are approaches which used only 1 classifier to perform the recognition, while others used several, one for each category to identify its classes [153]. Hence, we trained 5 additional classifiers to compare the performance of the recognition system. One classifier is trained using the Cat\_CNN architecture described in Table 4.1 to identify the category of the detected sign, and four more classifiers using the Class\_CNN architecture (Table 3.1) to recognize the classes for each category. In such way, the output of the category classifier (4 categories + background) filters out false positives and defines the input for the corresponding class classifier. The four class classifiers : Danger - 15 classes, Prohibitory - 12 classes, Mandatory - 8 classes and Others - 8 classes, (all of them adding the background class +1) are responsible for identifying the specific class in the respective category. The training procedure follows the same standard as described for the GTSRB (train for 40 epochs without data augmentation and use the weights as initialization to train it again for 100 epochs using data augmentation).

#### 4.4.3/ SEMANTIC SEGMENTATION EVALUATION

We quantify the performance of the four semantic segmentation models described in Section 4.3.1 with the UTBM-2 dataset to perform environment perception on French urban scenarios.

The same training procedure, described in Section 4.4.2.1, was applied for all the semantic models. This in order to be able to compare the models performances when the dataset, training capabilities and training parameters are standardized.

Several metrics are used for evaluation, the most common ones reside on pixel accuracy and Intersection over Union (IoU). Pixel accuracy is the simplest metric as it only expresses the ratio between the amount of correctly classified pixels over the total number of them. It is mostly presented as a global measure and/or in a per class manner when it is averaged over the total number of classes. It is computed with the following equation :

$$mPA = \frac{1}{k+1} \sum_{i=0}^k \frac{p_{ii}}{\sum_{j=0}^k p_{ij}} \quad (4.1)$$

Where  $k$  represents the number of classes,  $p_{ii}$  the number of pixels of class  $i$  inferred to belong to class  $i$  (true positives) while  $p_{ij}$  and  $p_{ji}$  represent false positives and false negatives respectively.

Even though mPA metric seems to be good, it is biased towards the classes that occupy bigger portions in the image. For this reason, the mean IoU (mIoU) came into play for a fair class evaluation and it is the most preferred metric for semantic segmentation evaluation. It computes the ratio between the intersection (true positives) and the union of two sets (true positives, false negatives and false positives). Normally it is computed in a per-class basis and then averaged.

$$mIoU = \frac{1}{k+1} \sum_{i=0}^k \frac{p_{ii}}{\sum_{j=0}^k p_{ij} + \sum_{j=0}^k p_{ji} - p_{ii}} \quad (4.2)$$

For each model, we inspected the validation losses at each epoch and found out that the models were learning correctly until they reached 100 epochs. Hence, we used the weights of the last epoch to evaluate the models on the UTBM-2 test set. Table 4.2 shows the comparison results for each model. We decided to include the number of parameters,

TABLE 4.2 – Quantitative comparison of deep networks for semantic segmentation on the UTBM-2 dataset. The training time corresponds to 100 epochs while the testing time to the average per image.

Model	Training			Testing			
	Parameters	Data augmentation	Time	Time (s)	Global PA	Class mPA	mIoU
SegNet	34,970,214	Yes	13 hrs 50 min	0.2814	92.85%	57.34%	0.5072
PSPNet	56,002,278	Yes	10 hrs 45 min	0.1685	93.36%	61.69%	0.5521
DeepLabV3	47,957,184	Yes	9 hrs 26 min	0.0899	93.85%	64.37%	0.5844
BiSeNet	47,768,080	Yes	9 hrs 45 min	0.1106	95.74%	69.07%	0.6473

training and testing time together with the metrics mentioned before for evaluation. In this way, we are able to compare quantitatively not only their performances on the test set, but also the models' complexity relating the training and testing times. The test time presented corresponds to the average time obtained to segment one image in GPU mode. From Table 4.2, we can see that the best model regarding pixel accuracy and IoU is BiSeNet [186]. The fastest one predicting an image output is DeepLab [175] followed by BiSeNet. Even though SegNet seems to be the simplest model with the 34.9 millions of parameters, the training and testing times are the longest compared to the other three methods. The reason behind this behavior could be due to the deconvolutional and up-sampling layers, as the model has to pass through the same number of convolutional blocks to recover the input image size. PSPNet [170], on the other hand, is the most complex model, but due to the pyramid strategy, it is able to perform training in a considerable amount of time comparing it to the others. Regarding the results between DeepLab and BiSeNet, the models' complexities are similar considering their training and testing times, but their performances vary significantly. BiSeNet outperforms DeepLab with 2% in the global PA, around 5% for the Class mPA and 0.06 for mIoU. Quantitatively, the mIoU results could appear to be low, and a per class evaluation will be useful to know in which specific classes the model is having troubles.

Table 4.3 indicates the mIoU results per class for the four models. Analyzing these results, we can see that all models have troubles segmenting small object classes like lane markings, crosswalks, humans, traffic signs, traffic lights, bicycles and poles with less than 0.6 IoU, while succeeding for the frequent big ones. This behavior is not surprising because classes like buildings, sky, road, appear in every single image, and they occupy a considerable percentage of the total image size. Consequently, due to the class imbalance, the models are able to learn the classes with higher number of pixels better than for the ones with small representations (refer to Fig. 2.12).

TABLE 4.3 – mIoU quantitative comparisons of semantic segmentation models on the UTBM-2 test set for 15 classes.

Model	building	pole	sky	vegetation	road	laneMarkings	parkingBlock	sidewalk	crosswalk	human	refugeeIsland	vehicle	bicycle_motorcycle	trafficLights	trafficSigns	mIoU
SegNet	0.7674	0.2502	0.8939	0.8829	0.9388	0.4781	0.7360	0.8254	0.4724	0	0.3459	0.7353	0	0	0.2823	0.5072
PSPNet	0.7942	0.3593	0.9150	0.9033	0.9360	0.5002	0.6891	0.8252	0.2751	0.2524	0.3788	0.7719	0.0036	0.1711	0.5063	0.5521
DeepLabV3	0.8030	0.3797	0.9202	0.9034	0.9453	0.5045	0.7379	0.8516	0.4891	0.2183	0.5749	0.7980	0.0607	0.1538	0.4254	0.5844
BiSeNet	0.8459	0.4420	0.9494	0.9342	0.9624	0.6266	0.8459	0.8949	0.5741	0.3240	0.6662	0.8581	0	0.2170	0.5683	0.6473

Besides quantitative comparisons, qualitative ones are also necessary to evaluate the models performances regarding boundary distinctions. Fig. 4.11 shows some example results performed on the UTBM-2 test set. SegNet results present smooth boundary transitions between classes, but it is not capable to distinguish the boundaries of small objects like traffic signs. It also has troubles detecting human class and traffic lights (Table 4.3). For PSPNet, the image results show that the model detects better small classes like traffic signs, traffic lights and humans, but, for some other classes like road, vehicle, sidewalks, it has difficulties in learning the context information since it is not able to differentiate them properly. The same behavior is present for DeepLab results, as most of the classes contain holes (small pixel portions with a different class than its surroundings). The interesting fact is that in its definition, the model uses CRF to improve boundaries, but for some images, it does not seem be working properly. Regarding BiSeNet, the image results show that, its context path definition is able to distinguish the classes properly. Small size objects are segmented more accurately together with their boundaries.

Quantitatively and qualitatively, BiSeNet outperforms the other models. It is able to segment with at least 0.85 IoU the following classes of interest : road, sidewalk, parking block, vehicle. Nevertheless, for some other important classes like humans, traffic signs, traffic lights and crosswalks, its performance is less than 0.6 IoU. Even though BiSeNet is not perfect, it is the one that we will use for environment perception as it provides the best results and its execution time is considerable for this kind of task.

Regardless of the semantic segmentation method chosen, individual object distinction is not possible with this kind of approach. For this reason, a complementary module is needed to distinguish and classify traffic signs.

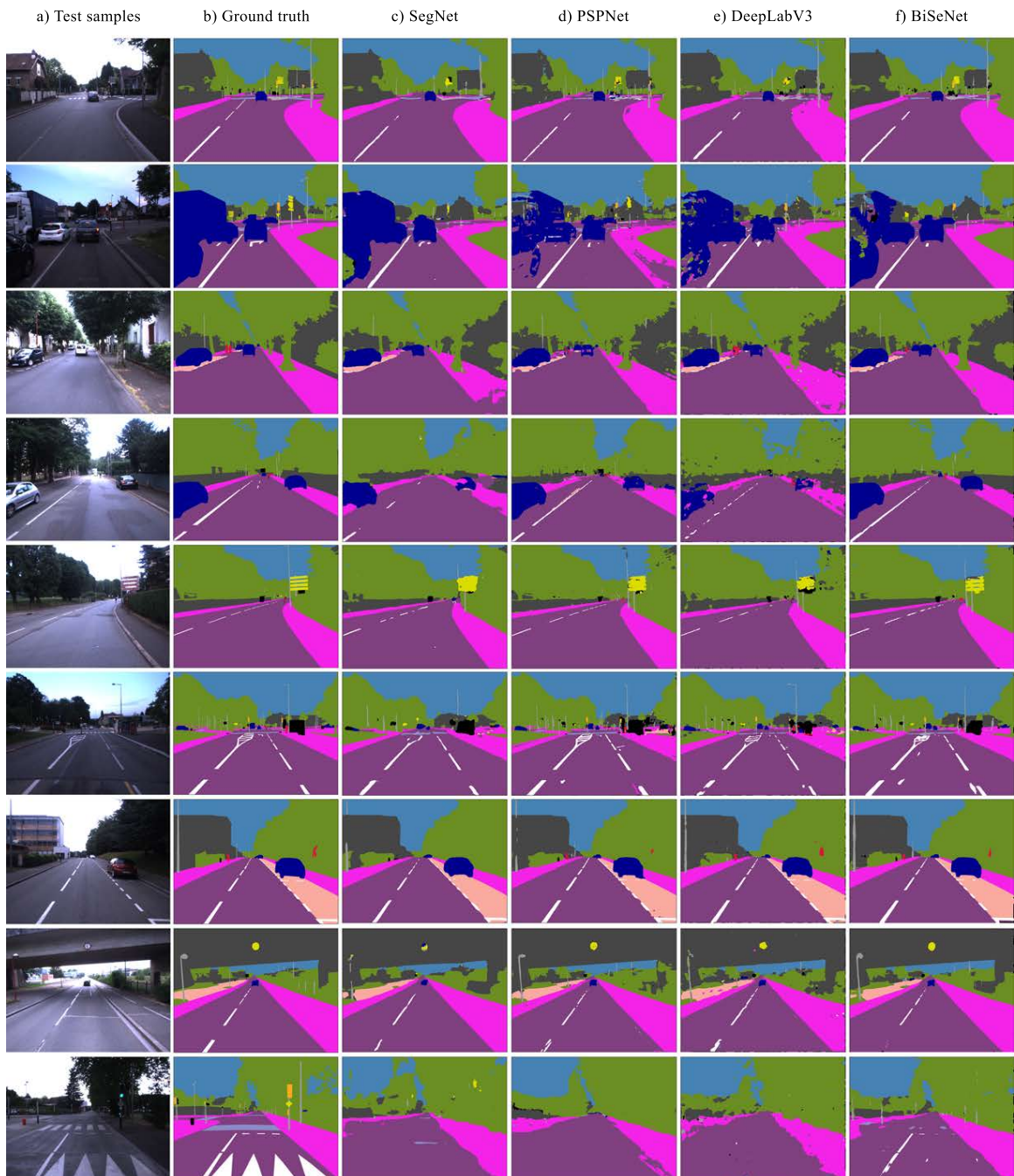


FIGURE 4.11 – Semantic segmentation results on test samples of UTBM-2 dataset.

#### 4.4.4/ TRAFFIC SIGN RECOGNITION EVALUATION

Because the detector module (Mask R-CNN) identifies a unique class for traffic signs, the category identification is performed further with the classification module. Hereupon, we will discuss first the classification results in order to continue with the final detection performance in Section 4.4.4.2.

TABLE 4.4 – Classifiers' accuracy results and characteristics trained on the proposed Class\_CNN and Cat\_CNN described in Section 4.3.2.2.

Model	Classifier name	No. Classes	Data augmentation	Parameters	Accuracy
Cat_CNN	Categories	5	Yes	170,621	99.92%
Class_CNN	Danger	16	Yes	2,081,712	99.25%
Class_CNN	Prohibitory	13	Yes	2,080,941	99.76%
Class_CNN	Mandatory	9	Yes	2,079,913	99.89%
Class_CNN	Unique	9	Yes	2,079,913	99.64%
Class_CNN	All	44	Yes	2,088,908	99.55%
Class_CNN	Urban	133	Yes	2,111,781	99.07%

#### 4.4.4.1/ CLASSIFICATION PERFORMANCE

As seen in Chapter 3 our proposed Class\_CNN provides good stability and trade off between model complexity (number of parameters) and recognition accuracy compared to the state of the art. Moreover, we inspected the running time and confidence of our propose network. Our model is capable of classifying in our PC, in GPU mode, a traffic sign in 0.16 milliseconds which makes it optimal for the whole pipeline of traffic sign recognition. Evaluating its performance, our Class\_CNN is capable of classifying correctly 99.38% traffic signs with a confidence probability bigger or equal to 0.9. For this reason, this threshold was chosen for our pipeline (see Fig. 4.7) to predict or discard a detected traffic sign. In this way, we make sure the output of the proposed traffic sign recognition module is accurate and reliable for driving vehicles.

Adding the background class, the 7 classifiers comprising Class\_CNN (Table 3.1) and Cat\_CNN (Table 4.1) are trained 5 times and only the top results are presented in Table 4.4 together with their characteristics. As a reminder, the first six classifiers are trained on the GTSRB including a background class, while the last one called 'Urban', is trained on the subset chosen from the ETSD comprising as well the background class. If we compare the accuracy performance of the Class\_CNN before including the background class, we have obtained 99.61% (top accuracy), while the current performance is 99.55%, dropping by 6%. This behavior is normal since the background class contains random characteristics making it hard for any classifier to learn properly the representative features. On the other hand, comparing the classifier called 'All' with the 'Categories' one, we can see in Table 4.4 that the accuracy performance is better with the Cat\_CNN (99.92%) than with the Class\_CNN (99.55%) despite the  $\times 12.23$  less number of parameters. This performance is quite surprising but we will analyze in the next section the results obtained once

the detector (Mask R-CNN) passes its outputs to the classifier.

Including other category types, the 'Urban' classifier trained with Class\_CNN, decreases accuracy performance as text traffic signs are the hardest ones to recognize. Nevertheless, its accuracy remains above 99% which makes it a reliable classifier even with text and rectangular traffic signs.

The traffic sign recognition module is evaluated together with the detection module (Mask R-CNN) and all the classifiers from Table 4.4.

#### 4.4.4.2/ DETECTION PERFORMANCE

Here, we evaluate the detection performance with Mask R-CNN and several classifiers. The first evaluation is made for symbol-based signs on the GTSDb to detect 4 Categories (43 classes) with 2 classifiers : one which is only capable of detecting the categories (Cat\_CNN Categories from Table 4.4) and another one capable of performing the class identification (Class\_CNN All from Table 4.4). The second evaluation is performed for symbol and text-based signs with two datasets. The first one is the proposed extended version of the GTSDb and the second dataset is UTBM-2. Both are evaluated to detect 8 categories (132 classes) of traffic signs (Class\_CNN Urban classifier from Table 4.4) : danger, priority, prohibitory, mandatory, special regulation, direction, additional panels, and others.

In both cases, a correct detected traffic sign is considered if its Jaccard similarity coefficient (Intersection over Union - IoU) is at least 0.5. The IoU is computed following Equation 4.3.

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \quad (4.3)$$

When the IoU is equal or bigger than 0.5, it counts as a True Positive (TP), otherwise as a False Positive (FP). The traffic signs which are not detected by Mask R-CNN are treated as False Negatives (FN).

The evaluation of detection approaches is performed most commonly with Precision-Recall values and the area under this curve (AUC). Precision indicates how good the model is on the detected traffic signs and it is computed with Equation 4.4. Recall, on the other hand, refers to how many good detections were actually identified according to the ground truth (Equation 4.5). Most of the related works that evaluated their detectors on



the GTSDb, only considered the AUC and Recall values. Hence, we will use the same metrics for these detectors.

$$Precision = \frac{TP}{TP + FP} \quad (4.4)$$

$$Recall = \frac{TP}{TP + FN} \quad (4.5)$$

Initially, and in order to compare the detection performance with other methods, we evaluate Mask R-CNN on the GTSDb using the fine-tuned weights obtained at epoch 46 to detect the original 4 Categories of the German symbol-based traffic signs. As Mask R-CNN only predicts if the detected RoI is a traffic sign, a lot of objects, including false matches, are passed to the Refinement module to perform some filtering and then pass its output to the classifier. This filtering process is carried out keeping only the detected Rols whose confidence probability (class score) is equal or bigger than 0.8 (threshold set manually). As the class probability of a positive RoI does not influence in the detection, it gives a good notion of the actual traffic sign since negative or false matches tend to have low confidence probabilities. After this process is done, the inputs (filtered detected Rols) are preprocessed as described in subsection 4.4.2.3. Next, the classifier receives the inputs which are not always traffic signs and has to be able to filter out the backgrounds while performing the category identification. This final filtering is made by the classifier, learning a background class together with the others.

The Cat\_CNN classifier, defined in Table 4.1 and trained with background class (first row in Table 4.4 and the Class\_CNN classifier, defined in Table 3.1 and trained as well with background class (Class\_CNN All in Table 4.4), are evaluated together with Mask R-CNN to detect 4 categories on the GTSDb test set. Fig. 4.12 shows the precision-recall curves obtained with an IoU of 0.5. For the danger (Fig. 4.12a), prohibitory (Fig. 4.12b) and others (Fig. 4.12d) categories we can see that both CNNs performed almost the same, while for the mandatory category (Fig. 4.12c), the category classifier (Cat\_CNN) has more troubles identifying the classes.

The detection performances are compared quantitatively with the state of the art in Table 4.5. Treating all traffic signs as one category, we obtained directly from Mask R-CNN, 99% recall and 86% precision. The recall value reached, allows us to confirm that Mask R-CNN serves as good detector as it only missed to detect 5 signs of the 361 in the test set. However, for the precision value, we can see that, even if it is good detecting most of

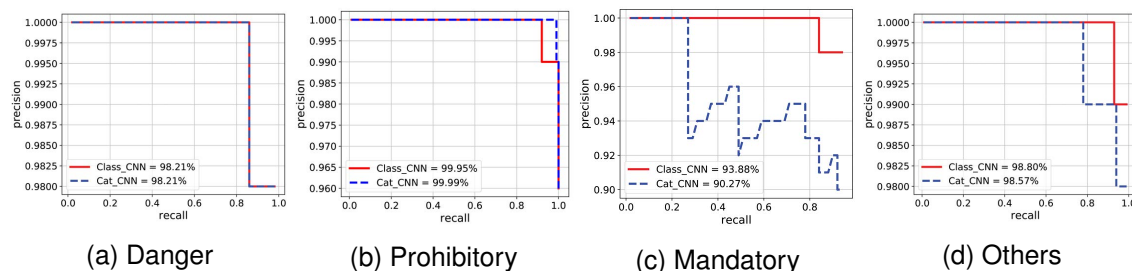


FIGURE 4.12 – Precision-Recall curves of the detection module (Mask R-CNN) using 2 CNNs for category identification. Cat\_CNN refers to the categories classifier defined in Table 4.1, while Class\_CNN refers to the classes classifier defined in Table 3.1.

TABLE 4.5 – Detection results on the GTSDb test set for 0.5 IoU (AUC values)

	Method	Danger	Prohibitory	Mandatory	Others
Traditional features	HOG+LDA+SVM [92]	99.91%	100%	100%	-
	ICF [87]	100%	100%	96.98%	-
	HOG+SVM [85]	98.85%	100%	92%	-
	ROI+HOG+SVM [88]	98.72%	99.98%	95.76%	-
	HOG+CNN [93]	99.73%	-	97.62%	-
	ROI+HOG+SVM [153]	97.13%	99.29%	96.74%	-
Automatically features	ROI+Multi-task CNN [182]	98.34%	99.99%	98.72%	-
	Faster-RCNN+CNN [181]	100%	96%	100%	99%
	2Dpose-boundary CNN [180]	99.73%	99.89%	99.16%	-
	modified YOLOv2 [169]	96.12%	96.81%	94.02%	-
	MaskRCNN+Cat_CNN ( <b>Ours</b> )	98.21%	99.99%	90.27%	98.57%
	MaskRCNN+Class_CNN ( <b>Ours</b> )	98.21%	99.95%	93.88%	98.80%

the correct traffic signs, it also detects several false matches (57). Nevertheless, after the classifier is applied, we are able to obtain a mean average precision mAP (considering the 4 Categories) of 96.76% with the categories classifier (Mask-RCNN+Cat\_CNN) and 97.71% with the classes classifier (Mask-RCNN+Class\_CNN). Such results let us affirm that, the classes classifier (Class\_CNN) performs better than the Cat\_CNN even if its classification accuracy is higher than learning all classes at once (Table 4.4). Both CNNs have troubles learning the mandatory category as most of the state of the art methods shown in Table 4.5. As the CNN with deeper layers (Class\_CNN) performed better, we will consider it as the final classifier for our proposed system.

Methods based on HoG [92, 87] features obtained almost perfect results (see Table 4.5), but these approaches are shape dependent, which makes them very specific for certain tasks. On the contrary, CNN based approaches are more suitable for general purposes,

TABLE 4.6 – Detection results on the extended GTSDB and UTBM-2 test sets using MaskRCNN+Urban\_Class\_CNN for 0.5 IoU (AUC values)

Category	Extended GTSDB	UTBM-2
Danger	98.41%	-
Priority	97.08%	69.44%
Prohibitory	96.12%	62.06%
Mandatory	93.50%	8.33%
Special regulation	70.60%	67.68%
Direction	66.02%	60.23%
Additional panels	80.14%	-
Others	67.55%	90.91%

putting aside the tedious and hard work to choose the right features for specific shapes. Comparing our results with the state of the art methods that use CNNs, our proposed pipeline (Mask R-CNN + Class\_CNN) is very competitive in terms of traffic sign detection including the Others (Unique) category that many studies did not take into account. Only the authors in [181] (Faster-RCNN+CNN) considered this last category and reported very high AUC values in their results. However, they mentioned to have achieved a mAP of 84.5% and recall of 97.81%, which make us believe that their results reported are not actually AUC values, but instead recall ones. For this reason, we discarded the method Faster-RCNN+CNN [181] for comparison. From the methods that extract features automatically, only [182] and [180] perform slightly better than ours (around 1.3% and 1.8% mAP respectively) which makes our method competitive.

Furthermore, and in order to evaluate other important traffic signs, we performed detection on the proposed extended version of the GTSDB and UTBM-2 datasets and analyzed their results. Mask R-CNN was run using the fine-tuned weights for both datasets, together with the Class\_CNN classifier trained on the selected urban/rural subset of the ETSD (Urban Class\_CNN from Table 4.4).

Regarding the extended GTSDB dataset, we obtained 83.68% mAP, 81% precision and 83% recall; while for UTBM-2 dataset a mAP of 59.78% was accomplished, 76% precision and 77% recall. These results show that, the UTBM-2 dataset is more challenging than the extended GTSDB. Considering only the German roads, we can see that when including other than symbol signs, the detector missed 140 signs (FN) and detected incorrectly 168 ones (FP). The AUC values for each dataset and category can be seen in Table 4.6.

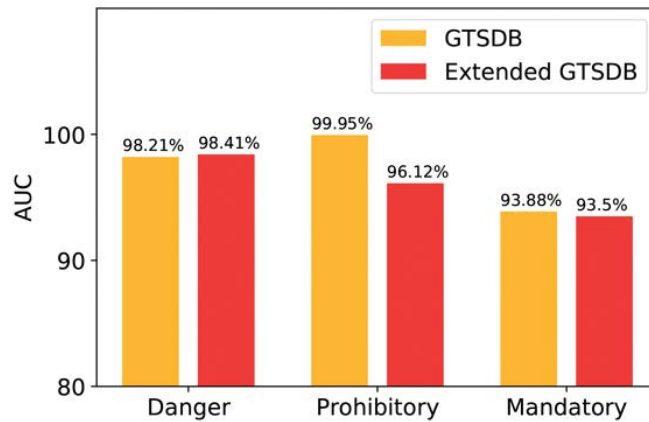


FIGURE 4.13 – Comparison between AUC values of the common categories between the GTSDB and its extended version.

According to Table 4.5 and Table 4.6, the detection rate for the danger category in the extended GTSDB, increased by 0.2% from the original GTSDB (see Fig. 4.13), even when the number of danger sign classes is higher in the urban set. The AUC value for the prohibitory and mandatory categories dropped by 3.79% and 0.38% respectively (Fig. 4.13). The reason behind this behavior is due to the variety of classes on the urban set and the sizes of the signs labeled. In the original GTSDB, only the most close and distinguishable signs are considered, while for the extended version, we labeled all signs visible and recognizable by human eye. The rest of the categories in the extended GTSDB performed poorly (below 90%) because the classes are harder to recognize due to the very different aspect ratios they possess and to the fact that they include text. By nature these signs are very difficult to recognize.

Since the Class\_CNN outputs the label of the specific sign, and from this label we infer it's category, Fig. 4.14 and Fig. 4.15 illustrate some examples of the results obtained by our traffic sign recognition module on the GTSDB and UTBM-2 datasets respectively. The traffic sign accuracy indicated in green/red text at the bottom left in every image, corresponds to the classification accuracy obtained for the detected signs only in that image.

For French roads, Fig. 4.15 illustrates some examples where the recognition module is able to performed very good  $\approx 100\%$  (left column in Fig. 4.15), but also, poorly for some other images (right column in Fig. 4.15). Judging visually, for UTBM-2 dataset, the module fails to recognize very small traffic signs, confuses other objects with signs, like street name panels, and images with very different lighting conditions (over saturated light sensor) are some of the problems that encountered the recognition module.

TABLE 4.7 – Classification evaluation for German and French roads with the GTSDDB, the extended GTSDDB and UTBM-2 test sets. The evaluation is performed on the detected signs extracted by Mask R-CNN + the Urban Class\_CNN.

Category	Original GTSDDB			Extended GTSDDB			UTBM-2		
	Detected	Correct class	Accuracy	Detected	Correct class	Accuracy	Detected	Correct class	Accuracy
Danger	62	62	100.00%	59	59	100.00%	-	-	-
Priority	-	-	-	75	72	96.00%	17	10	58.82%
Prohibitory	165	162	98.18%	202	187	92.57%	41	33	80.49%
Mandatory	45	45	100.00%	63	53	84.13%	1	0	0.00%
Special regulation	-	-	-	58	48	82.76%	18	18	100.00%
Direction	-	-	-	124	101	81.45%	71	59	83.10%
Additional panels	-	-	-	44	39	88.64%	-	-	-
Others	88	87	98.86%	76	63	82.89%	9	9	100.00%
All	360	356	98.89%	701	622	88.73%	157	129	82.17%

Nevertheless, for the GTSDDB (Fig. 4.14), the module did not encounter the same issues as with UTBM-2, and in consequence its performance is higher. In Fig. 4.14, the column on the left represents results computed on GTSDDB recognizing 43 classes (4 Categories), while the column on the right, shows detection and classification results obtained on the extended version of GTSDDB recognizing 132 classes from the urban set extracted from the ETSD [184]. In the extended version, we are able to detect and classify directional signs, obstacle signs, additional panels as well as special regulation signs. Hence, a more reliable interpretation of the environment for driving scenarios can be made recognizing all traffic signs with our proposed traffic sign recognition module.

Additionally, we provide classification evaluation on the traffic sign proposals detected by Mask R-CNN using the Class\_CNN trained with the urban set of the ETSD [184]. The classification evaluation is performed on the GTSRB and UTBM-2 datasets. Firstly, we evaluate the recognition on GTSDDB and its extended version (see Table 4.7). For the GTSDDB we have contemplated 360 signs after filtering processes (detection class score  $\geq 0.8$  and classification probability  $\geq 0.9$ ) from which only 356 are classified correctly obtaining a classification accuracy of 98.89%. In the extended GTSDDB, 701 signs were detected, where 622 are predicted correctly resulting in 88.73% accuracy. It should be noted that the Others category does not include the same classes in both datasets (refer to Fig. 3.9 in Chapter 3) but are displayed like that in order to match the naming convention. Fig. 4.16 and Fig. 4.17 show some examples of the misclassified detected signs for the GTSDDB and its extended version respectively. In Fig. 4.16, the detected signs are not labeled in the ground truth indicating that their label is background class (GT :43). For Fig. 4.17, we have the same issue. Even if some signs seem to be traffic signs, because they were not recognizable by human eye, we did not label them.

Referring to the classification evaluation of UTBM-2 on the detected traffic signs by Mask R-CNN (see Table 4.7), we have contemplated 157 signs after the filtering process (detection class score  $\geq 0.8$  and classification probability  $\geq 0.9$ ) from which 129 signs are classified correctly obtaining a classification accuracy of 82.17%. Some misclassified signs from the UTBM-2 test set are illustrated in Fig. 4.18. There we can see that, the traffic signs are not recognizable and mostly, as they are not labeled in the dataset, their ground truth becomes automatically background class (GT : 132).

Yang et al. [153] reported 98.24% accuracy recognition rate after all the signs were detected on the GTSDb. Comparing their method to ours (98.89%), we surpass their performance with 0.65% accuracy more. Other studies do not provide the recognition rate after the detection module. For this reason it is not possible to compare with them. Along with this accuracy, we evaluated the classification rate on the detected signs per image, as shown in the examples of our recognition module (Fig. 4.14 and Fig. 4.15). Then we averaged the results, obtaining on the GTSDb test set (300 images) 99.39% with MaskRCNN+Class\_CNN (43 classes + background), 89.97% with MaskRCNN+Urban\_Class\_CNN (132 classes + background) on the extended GTSDb test set, and 80.19% with MaskRCNN+Urban\_Class\_CNN (132 classes + background) on UTBM-2 test set (109 images). Since the complexity of each dataset and every image varies a lot, this type of evaluation is useful if individual outputs are required giving the same weight to each image.

The running time is not considered in this study for comparison purposes, because the time reported by each approach depends on the hardware capabilities where they were tested. Nonetheless, we can give the reader an estimation of the running time of our traffic sign recognition module tested in our PC in GPU mode. For an image size of  $1280 \times 960$  pixels, it takes approximately 0.32 seconds to detect and classify all traffic signs. In other words, our system is capable of running at  $\sim 3.3fps$  which makes it relatively a good choice for real-world applications.

Interestingly, no matter the difficulty of some categories, like additional panels, special regulation and, others; the recognition system needs to consider them to interpret correctly the driving rules in a similar way to how humans do. For this reason, our proposed system is a good choice for Driver Assistant Systems (ADAS) and autonomous vehicles.

## 4.5/ CONCLUSIONS AND FUTURE WORKS

In this Chapter, we proposed a system composed of two modules to perform environment perception and traffic sign classification. The first module makes use of a semantic segmentation approach for outside scene understanding. For this module, we have analyzed four CNN architectures and compared them between each other to reveal the practical trade-offs between training time, memory, and accuracy. The comparative study is performed on the proposed UTBM-2 dataset to accomplish environment perception on French urban scenes.

The second module, proposes a symbol and text-based traffic sign recognition system for European urban environments. The detection part (Mask R-CNN) identifies possible traffic signs while at the same time segments a mask for each of them. These masks serve as a localization refinement avoiding other post processing tasks, like shape estimation, to provide an accurate RoI. The refinement procedure is introduced to discard objects that may not be traffic sign and pre-process the inputs before passing them to the classifier. The classification part, performed with a proposed CNN, filters out the RoI inputs which are background and recognizes the specific classes and categories of the traffic signs.

The proposed Class\_CNN architecture obtains a top accuracy of 99.61% on the GTSRB test set, while at the same time, its learning process is very stable using Dropblock as regularization technique. The performance of our CNN architecture provides a good trade off between the number of parameters, accuracy and computational time compared to other methods in the state of the art, which most of the time, use an ensemble model with several CNNs.

A urban/rural subset subtracted from the ETSD [184], allowed us to consider a complete dataset with symbol and text signs dealing at the same time with intra-class variability from 6 countries (Belgium, Croatia, France, Germany, Netherlands and Sweden). Such characteristics are of vital importance for a system to recognize properly the same class, even if small variations are presented when driving from one place to another, or changing countries.

Both modules work with an image size of  $1280 \times 960$  to avoid information loss from small objects (traffic signs) when resizing the images. The environment perception module (first module) is able to detect with at least 0.85 IoU classes of interest like road, parking block, sidewalk, and vehicle, while performing poorly on the ones with small objects. For this

reason the second module was introduced to recognize these kind of objects, specifically traffic signs.

The traffic sign recognition module provides an overall recall detection rate of 99% on the GTSDDB test set with 98.89% accuracy classification on the detected signs and an average accuracy (accuracy per image) of 99.39%. For the proposed extended version of the GTSDDB, it obtains 83% recall detecting symbol and text signs while achieving 88.73% accuracy when recognizing them, and 89.97% accuracy as an average (evaluating images individually). Whereas for the UTBM-2 dataset, it reaches 77% recall rate for symbol and text signs recognizing the detected ones with 82.17% accuracy and 80.19% in average (109 images). Detection and classification dropped on the extended GTSDDB version due to the nature of the classes it contains. The most challenging ones are based on text and with very different aspect ratios (belonging to Directional and Additional panels categories).

By nature, small signs like Additional panels are hard to detect, which makes them the most difficult ones for detection approaches as the portion they occupy in the image is relatively small compared to other objects in the scene. At the same time, text-based signs are confused with other objects containing text, like advertisements, store signs or street name panels due to the variability of appearances they have.

In future work we intent to : 1) Establish a based CNN architecture to extract a common feature map and use it for both modules 2) take into account data augmentation for instance segmentation in order to increase the detection performance of Mask R-CNN, since the training sets of the GTSDDB and UTBM-2 datasets are quite small (600 and 482 images respectively) ; 3) train other detectors based on CNNs to evaluate the running times and compare their performances ; 4) apply GANs like authors in [179] did, to generate features for the very small traffic signs similar to the ones of clear and big signs, this in order to improve our detection and recognition.

Henceforth that we have a system to perceive the environment, in the following chapter we will focus on the second problem of this thesis work : path tracking. In a typical automated driving system, the path to follow and speed are already defined based on what was perceived in the environment, the vehicle localization and the path planning module. In our case, the reference path is provided as a priori information, analyzed and passed to the vehicle control. The speed will be computed based on an off-line analysis of the visual information. Thus, the vehicle control will have the necessary data to calculate the steering angle for tracking successfully the reference path.





(a) GTSDDB

(b) Extended GTSDDB

FIGURE 4.14 – Example of detection and classification results by our proposed traffic sign recognition module on the GTSDDB. Column (a) shows some images evaluated on the original GTSDDB using Mask R-CNN + Class\_CNN trained on the GTSRB to detect 43 classes. Column (b) refers to the detection with Mask R-CNN on the extended version of the GTSDDB + Urban\_Class\_CNN trained in the urban set of the ETSD to detect 132 classes.



FIGURE 4.15 – Example of detection and classification results by our proposed traffic sign recognition module on UTBM-2 dataset. The recognition is performed with Mask R-CNN + Urban\_Class\_CNN trained in the urban set of the ETSD to detect 132 classes. The left column shows image examples with recognition accuracy above 75% while the right column shows images with traffic sign recognition accuracy under 75%.

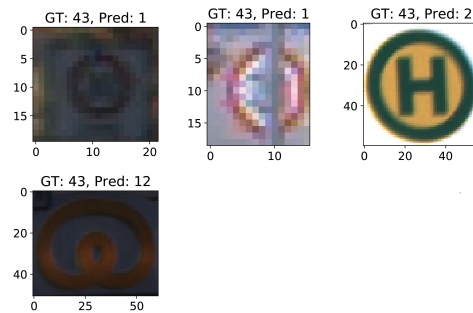


FIGURE 4.16 – Incorrect predictions after detection performed on the GTSDb with the Class\_CNN trained on the GTSRB.

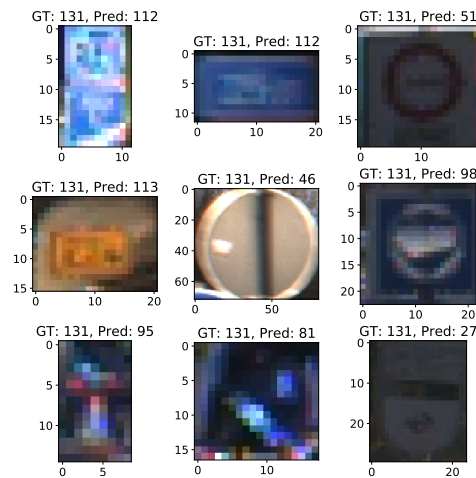


FIGURE 4.17 – Incorrect predictions after detection performed on the GTSDb extended version with the Urban\_Class\_CNN trained on the selected urban set of the ETSD.

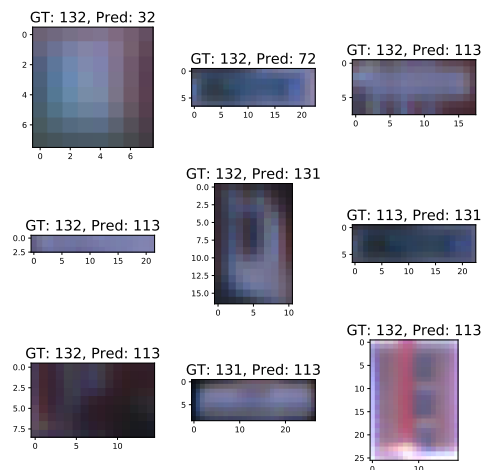


FIGURE 4.18 – Incorrect predictions after detection performed on the UTBM-2 dataset with the Urban\_Class\_CNN trained on the selected urban set of the ETSD.

# DYNAMIC SPEED ADAPTATION SYSTEM FOR PATH TRACKING BASED ON CURVATURE INFORMATION AND SPEED LIMITS

## 5.1/ INTRODUCTION

Autonomous vehicle navigation has been a challenging field of study where today the main goal is to provide safety. In order to accomplish that, vehicle control has to perform accurate path tracking; in other words, it should minimize the lateral distance between the vehicle's position and the defined path.

In real driving scenarios, speeding is one of the main causes for traffic accidents [66] which has been considered for Advance Driver Assistance Systems (ADAS). The correlation between speeding and lateral errors is positive, and in consequence, safety related issues will depend on vehicle's speed control.

Intelligent Speed Adaptation (ISA) systems have proven in several studies [23, 31, 37, 74] to reduce accidents by respecting speed limits. However, there are several other factors that need to be considered for precise speed control. One of the major ones involves road geometry.

Analyzing road structure will permit the identification of straight and curved segments ahead while driving, allowing the lateral control system to adjust the vehicle's speed accordingly. Curve detection is of vital importance since crash rate is at least 1.5 times higher than in tangent (straight) segments [75]. In addition, when a curve becomes sharper, the number of accidents increases [71, 3]. Hence, different studies have focused on analyzing how processing road structure [33, 84, 124] may benefit control systems.

Using speed limit information, systems like ISA will inform the driver when exceeding speed limits. On the other hand, approaches considering curvature information [33, 84] will emit warnings when approaching the curve too fast. In any case, control systems require precise knowledge of the vehicle's global position. This location information is usually provided by Global Positioning Systems (GPS), but its main drawback is that it suffers from big positioning errors. These errors are caused by different factors like blockage, multipath, etc., specially in urban environments. In order to overcome these big errors, different solutions proposed the use of additional sensors such stereo vision, IMU, radar, or LiDAR [124, 28, 67, 100]. For example, vision-based solutions together with digital maps, are capable of estimating the ego motion of the vehicle and correct the vehicle's global position using a map-matching algorithm [67]. In [100] the use of cameras, accurate digital map, GPS and inertial measurements improve the ego-localization of the vehicle using a variant of Kalman filter [6]. The main limitation of vision approaches is visibility, since shadows, highlights, occlusions, or weather conditions affect the accuracy of data analysis. Moreover, IMU sensors are able to estimate the current position of the vehicle. Fusing GPS and IMU with typically Kalman filter [6], provides absolute position and orientation, even if GPS data is not available all the time [152].

Road map databases (GIS maps) together with GPS data provide accurate road geometry and localization [75, 43]. An approach to analyze road geometry identifying critical curves where accidents occur frequently was proposed in [75]. Wang et al. [43] calculated road ahead of the vehicle through a flexible road model. Nevertheless, GIS information is not always available and GPS is starting to be accessible or will be soon for the general public. For this reason, the choice of GPS data for tracking approaches is convenient. The work presented here is based on real time kinematics GPS (RTK-GPS), which has the highest absolute position accuracy (up to a few centimeters), to implement an automatic approach that goes beyond warnings for adjusting the vehicle speed depending on the upcoming road characteristics, e.g. curves or speed limit zones.

In this Chapter, we propose a dynamic speed adaptation (DSA) method for control systems, which together with a speed limit database creation and curve extraction, adjusts automatically the vehicle's speed (see Fig. 5.1). These two approaches, speed limits database creation and curve extraction, are performed off-line including the preprocessing step. Their output provides the main parameters to be analyzed by the speed negotiation algorithm of DSA in order to compute the ideal speed. The identification of sharp curves is performed using GPS positions to define segments which belong to a curve or a tangent line. This in order to estimate the convenient driving speed for the detected high curva-

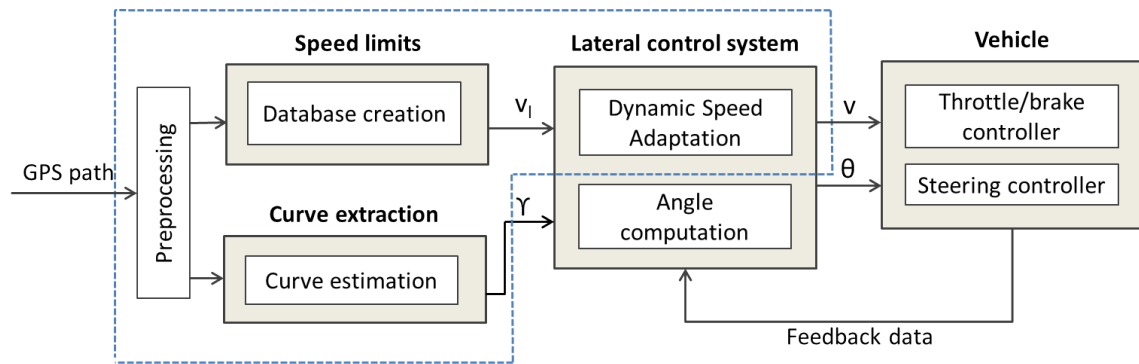


FIGURE 5.1 – Overall work-flow of the vehicle path tracking system. Blue dotted bounding box highlights our contributions.

ture segments. The speed limit database is created for the traveled paths identifying the positions where speed limits change. As the tracking performance is evaluated through lateral errors, our DSA is tested in simulations with different steering control algorithms. On the modeling level, our proposed work would :

- preprocess GPS path,
- extract position information of sharp curves and speed limit zones,
- compute recommended speed for each sharp curve,
- obtain the speed limits for the path,
- analyze the current vehicle position in the traveled GPS path,
- compute the triggered distance at which the vehicle needs to start decelerating to ensure smooth speed transitions,
- adjust speed during triggered distance to respect sharp curved speed or speed limits,
- control automatically vehicle's speed.

The rest of this Chapter is organized as follows. Section 5.2 presents a review of related works. Section 5.3 provides an overview of the steering control algorithms that we used to test our approach. Section 5.4 explains our DSA method. In section 5.5 we discuss and analyze the results obtained with and without considering automatic speed adaptation in the different steering control algorithms. The last section presents conclusions and highlights the main contributions to continue with future work directions.

## 5.2/ RELATED WORKS

As our DSA method estimates the speed based on either curvature information or speed limits, works related to these two problems are considered.

### 5.2.1/ INTELLIGENT SPEED ADAPTATION SYSTEMS

Intelligent Speed Adaptation (ISA) systems have proven to reduce the number of injuries and fatal accidents as shown in studies conducted in UK [23, 74], Sweden [15], Belgium [37], Denmark [38, 50], Australia [58], France [31], as well as in a study made by the European Union with the project DaCoTA [66].

Utilizing ISA systems does not only help to reduce tragic crashes or injuries but also : (1) gives the control system more time to identify and respond to hazards (unexpected events) [29] and (2) reduces the driver's need to monitor the speedometer with respect to external speed limit signs. For young drivers, it increases their safety since their inexperience of sharing their attention while driving and monitoring the speed to respect traffic rules could lead to accidents [58].

It is also known that in order to take advantage of fully controlled ISA systems, the speed limit information and the positioning system must be accurate [50]. For this reason, speed limit databases up to date with possible variations (time of the day, weather conditions, type of vehicle) are necessary. An approach to develop and use a speed limit database in ISA warning system was implemented as a mobile application software for Australian roads [149].

Even though Google maps application provides speed limits, it doesn't contain information regarding temporal situations (i.e. weather conditions, timing, construction areas) since it is an exhaustive work to maintain an updated database worldwide. Our speed limit database is based on static speed limits defined by the traffic speed regulations in France [12].

### 5.2.2/ CURVE SPEED ESTIMATION

The analysis of traffic accidents on curves has been a major concern of vehicle safety for several years [3]. No matter the efforts made to highlight dangerous curves with traffic signs [13], drivers are still surprised with the speed they need to reach to have an

appropriate control over the steering wheel.

Nowadays, in the era of autonomous navigation, various solutions have been proposed to try to overcome this problem. One of them focused on analyzing human behavior while driving in curves. Lehtonen et al. [105] examined how drivers anticipate their eyes look-ahead fixations on curves to provide visual guidance for steering. Zhang et al. [95] proposed a driver speed model for curved roads based on the recorded behavior of the driver.

Other works based their curve speed computation on the road shape estimation, either for implementing it in semi-autonomous or autonomous solutions. Related to road geometry analysis, identification of curved or straight segments has been performed either by analyzing vehicle's ahead motion [32, 41, 62], or by measuring physically related parameters like curve radius, length and angle [71, 75] using GPS and GIS information. Once these curves are identified, their corresponding speeds are calculated considering road friction and super-elevation angles [137, 33, 124].

Curve speed warning systems correspond to semi-autonomous solutions which consist in informing the driver, through sound or display, the recommended speed [44, 8, 33, 15]. E.g. Varhelyi [15] proposed a system which goes beyond curves, considering as well factors such as wet roads and visibility conditions (darkness) to compute the appropriate speed and inform the driver visually or audibly. Alternatively, the warning signal may come through some force applied to the throttle (accelerator) to prevent speeding. That is the case of Huth et al. [72], who proposed a curve warning system for motorcyclists through a force feedback throttle or a haptic glove.

Adaptive Cruise Control (ACC) systems, which are another type of semi-autonomous solutions, have also considered curve speeds together with speed limits to adjust velocity respecting headway distance to the lead vehicle [84].

Concerning autonomous solutions, it is not a surprise that the automotive industry is trying to develop control systems which consider every possible situation to drive safely. Nevertheless, user acceptance is still a factor to overcome in order to bring to reality autonomous vehicles [15, 58]. Through simulations, Park et al. [124] calculated curve speeds and applied them in a path tracking algorithm.

Even though curve speed estimation has been considered in several works, only few of them have been actually implemented and tested either in simulations or real environments [33, 84, 124]. Glaser et al. [33] as well as Park et al. [124] have computed the ideal speed for curves based on the analysis of geometric information, but no speed limits



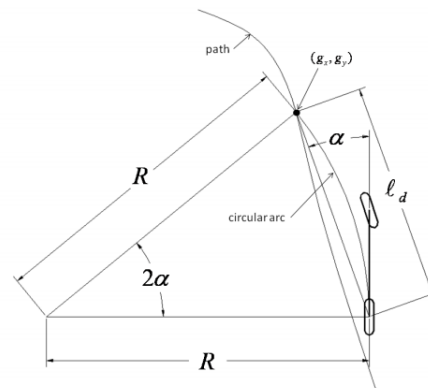


FIGURE 5.2 – Illustration of Pure Pursuit geometry [49].

have been contemplated. On the other hand, Lee et al. [84] considered speed limits and curve speeds, but their implementation is made for an ACC which is a semi-autonomous solution where the driver is able to take control of the vehicle and override appropriate speeds. Our work is targeted to be implemented as an autonomous solution for path tracking algorithms.

### 5.3/ STEERING CONTROL ALGORITHMS FOR PATH TRACKING

Path tracking has been one of the main challenges for autonomous navigation for the last 30 years. Its functionality resides in computing the appropriate commands for the vehicle to track a reference trajectory as accurately as possible. One of these commands is the desired vehicle orientation ( $\theta$ ) which together with speed ( $v$ ) will seek to achieve good performance.

This section provides a review of some steering control algorithms used in already tested autonomous vehicles. These methods will be evaluated in section 5.5 through simulations in order to prove that their performance can be improved significantly with our proposed Dynamic Speed Adaptation method.

#### 5.3.1/ PURE PURSUIT

Geometric based pure pursuit is one of the most basic and simple methods to compute steering wheel angle ( $\delta$ ) for a lateral controller. Its calculation relies on defining a goal point ( $g_x, g_y$ ) in a reference path according to a certain distance called look ahead distance ( $l_d$ ), to try to reach it at every time step  $t$  through a circular arc (see Fig. 5.2).

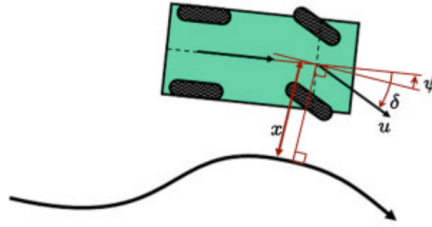


FIGURE 5.3 – Illustration of Stanley steering controller [28].

The pure pursuit control law is given as [49] :

$$\delta(t) = \arctan\left(\frac{2L \sin(\alpha(t))}{l_d}\right) \quad (5.1)$$

Where  $L$  is the distance between the front axle and rear axle (wheelbase) and  $\alpha$  is the angle between the vehicle heading vector and the look-ahead vector.

This algorithm suffers mainly of cutting corners (neglects curvature information) if look ahead distance is big, and of severe oscillations if it is defined too small.

### 5.3.2/ STANLEY METHOD

The Stanley steering controller was developed by the Stanford Racing Team and implemented in their autonomous vehicle (Stanley) for the DARPA Grand Challenge. Their controller is based on a non-linear feedback function based on lateral errors [28] measured from front wheel axle.

Four parameters are considered for the steering angle ( $\delta$ ) computation at each time step ( $t$ ) : lateral error ( $x$ ), vehicle speed ( $u$ ), a gain value ( $k$ ) and the angle difference ( $\psi$ ) between the vehicle and the nearest path segment orientation. An illustration can be seen in Fig. 5.3 and its equation is given by :

$$\delta(t) = \psi(t) + \arctan \frac{kx(t)}{u(t)} \quad (5.2)$$

This method is a simple basic approach which proved to be efficient in the DARPA race with an average lateral offset of  $\pm 74$  cm.

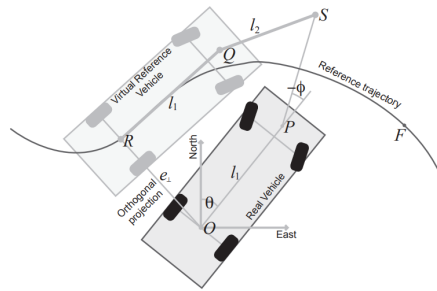


FIGURE 5.4 – Illustration of Alice steering controller [40].

### 5.3.3/ ALICE METHOD

The Caltech Team developed a control strategy for trajectory tracking based on a nonlinear control law [40]. The method was implemented in the Alice autonomous car which took part in the DARPA Urban Challenge.

The vehicle assumed to have Ackerman steer dynamics leading to use an approximation of the bicycle model. A graphical illustration is shown in Fig. 5.4 where the real vehicle is projected orthogonally onto the closest point of the reference path  $F$ . The rear axle center,  $O$ , is projected onto  $R$  and the yaw of the arising virtual vehicle is aligned with the tangent of the trajectory at  $R$ .

The steering wheel angle ( $\Phi$ ) is computed with the following formula at each time step  $t$  :

$$\tan(\Phi(t)) = \frac{-\cos(\theta(t))e_{\perp}(t) - (l_1 + l_2) \sin(\theta(t))}{l_1 - (l_1 + l_2) \cos(\theta(t)) + \sin(\theta(t))e_{\perp}(t)} \quad (5.3)$$

Where  $e_{\perp}$  is the lateral rear axle error (cross track error),  $\theta$  is the angle between the vehicle direction and the tangent of the projected vehicle position in the path (yaw error),  $l_1$  is the vehicle wheelbase and  $l_2$  is the distance to the target point.

The aim of this method is to reduce  $e_{\perp}$  and  $e_{\theta}$  which is equivalent to measuring the lateral error from the vehicle center line. For our purpose, the results presented in section 5.5 will show only the lateral error from the center of the car. This to provide a common comparison indicator when evaluating all the considered steering wheel methods.

### 5.3.4/ LOMBARD METHOD

Lombard et al. [144] proposed a steering controller based on the well known pure pursuit algorithm. Their method was implemented in a Renault Scenic car and showed at the

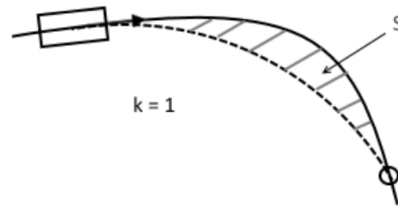


FIGURE 5.5 – Illustration of Lombard steering controller [144].

Intelligent Transportation System World Congress (2015).

The computation of the steering angle introduces a coefficient  $k$ , defined as  $1 - \alpha S$  (see equation 5.4), with the aim to reduce cutting corners effect without changing the target point distance or vehicle's speed. In other words, this coefficient tries to attenuate the area surface ( $S$ ) between the reference path and the arc drawn from pure pursuit (refer to Fig. 5.5). The steering angle parameter ( $\delta$ ) is defined as follows :

$$\delta = \arctan\left(\left(1 - \alpha S\right)\frac{E}{R}\right) \quad (5.4)$$

Where  $E$  is the distance between the axles of the car,  $R$  is the radius of the circle drawn to follow the target point, and  $\alpha$  is a coefficient set to 0.02.

Lombard's method provides lateral errors smaller than 1 m when the speed is set to 36 km/h.

## 5.4/ DYNAMIC SPEED ADAPTATION (DSA)

It is well known in path tracking literature that vehicle speed may increase or decrease lateral errors. In the ideal case, a good definition should lead the vehicle to track accurately the trajectory (with no errors) [80]. For this reason, most of the works have focused on defining a maximum speed (which is usually a small value [86, 124]) to tune the best performance with their algorithms. Nevertheless, speed variations in different situations need to be considered to simulate human driving behavior. The objective of our proposed DSA method is to simulate this behavior.

Our DSA approach defines the speed to be applied (through a speed negotiation algorithm) by the lateral control system in order to travel a safe trajectory. First, GPS information is preprocessed in order to remove noise and reduce the number of points. Second,

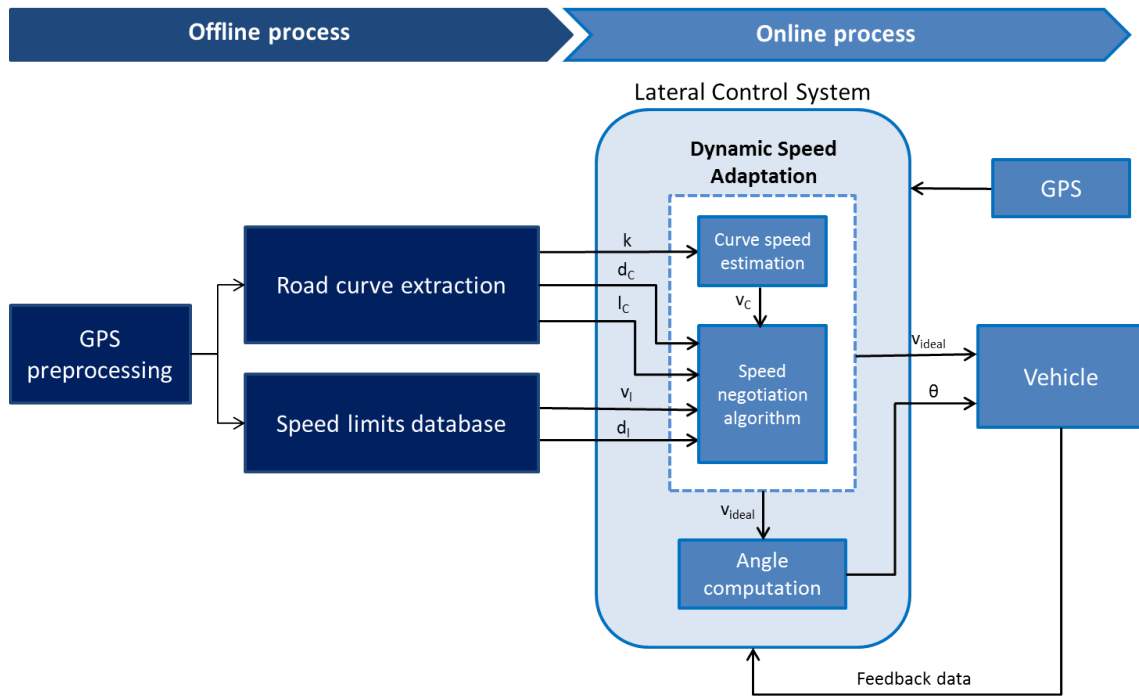


FIGURE 5.6 – Overall data-flow of the Dynamic Speed Adaptation module.

road speed limits together with their corresponding positions are detected in the traveled GPS path and saved in a database. Third, sharp curves are identified along with their positions, radius and length in order to compute their convenient speeds and send them to the speed negotiation algorithm (see Fig. 5.6). A more detailed description is presented in the following sub-sections.

#### 5.4.1/ GPS PRE-PROCESSING

Real Time Kinematic (RTK) refers to a technique able to correct positioning information (GPS) with centimeter level accuracy. This correction is carried out by a base station which reduces or removes positioning errors [42] caused by environmental surroundings or weather conditions.

RTK-GPS was used during data collections in order to have reliable GPS paths. Nevertheless, due to different circumstances like loss of connection between the base and receiver, noise appeared in the GPS data. This noise was removed manually in our datasets in order to have reliable learning routes. We identified the noisy points and interpolated their new positions based on the previous and following points.

Once GPS noisy points are removed, we computed the euclidean distance between each pair of points to keep GPS positions at approximately 3.5 meters distance between each

other. This with the aim to reduce the amount of data for further processing, i.e. road curvature estimation and speed limits extraction.

#### 5.4.2/ SPEED LIMIT DATABASE

Speed limits vary between countries, types of roads, areas, vehicles and inclusive timing [149]. They may also be affected by certain circumstances like constructions or special events around a zone (i.e. fairs, carnivals, street markets, etc.). Even though it is important to consider all possible scenarios for the speed limit database, it is very hard to maintain updated zones where temporary situations occur.

Digital maps contain speed limits information, but most of them are not free. Opens-treetmaps [39], on the other hand, provides open data with fixed speed limits (without considering temporary data), but for our traveled paths, the information is not included.

We created our own speed limits database, based on the typical manual driving situations and considering the code regulations established by the French government [12]. In France, urban areas have speed limits of 50 km/h varying to 30 or even 20 km/h in agglomeration areas, residential areas or school zones [12].

According to the recorded RTK-GPS information, we :

1. Define start and end points.
2. Identify positions where speeds limits change.
3. Compute distance ( $d_l$ ) from the speed limit change point to the reference starting point in the path.
4. Save distance ( $d_l$ ) for every speed limit changed position identified together with its respective speed limit value ( $v_l$ ).

We identified positions where speed limits change from 50 km/h to 30 km/h and vice-versa. By default the speed limit at the starting point is set to 50 km/h if no other speed limit is identified. An illustration of the procedure is presented in Fig. 5.7.

The creation of the speed limits database is performed off-line and only after GPS information is preprocessed. The description of the datasets and their speed limits identified is provided in section 5.5.

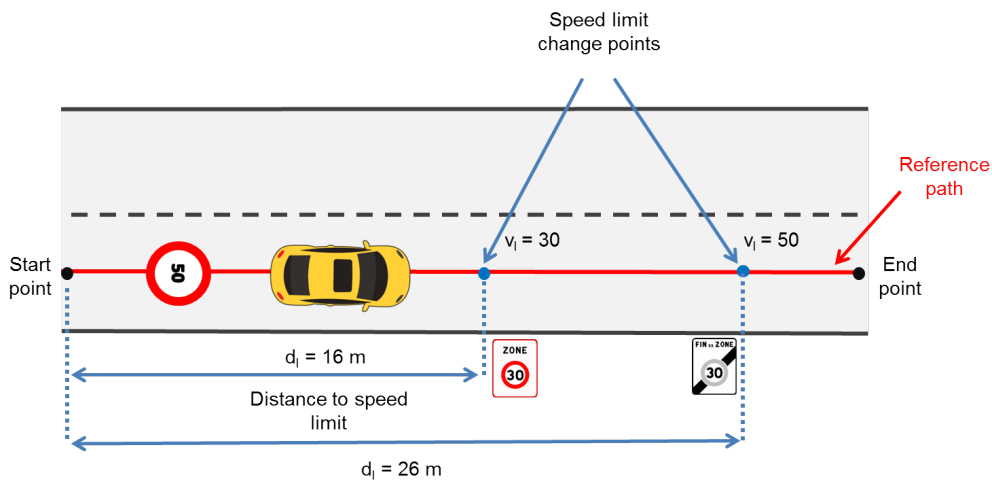


FIGURE 5.7 – Speed limit driving scenario

### 5.4.3/ CURVE SPEED ESTIMATION

Driving through an unknown sharp curve is always risky if the speed is not adapted to control the vehicle. In real life, the driver should be able to maneuver the steering wheel and reduce speed simultaneously in order to stay in the lane and pass the curve with comfort and safety. The same principle should be applied for autonomous control systems (reduce speed before going through a curve).

In order to control appropriately vehicle's speed, curve identification needs to be performed for the traveled GPS path to proceed with the computation of convenient speeds for each curve. Li et al. [75] extracted and classified curves considering road data from Geographic Information Systems (GIS). GIS store different types of information providing very accurate road geometry, but it is not always available. In our work, curvature analysis is performed with GPS data following the technique in [75] with minor modifications and extending its curve estimation to identify sharp curves.

Curve analysis is a very important step for intelligent vehicle systems [53] since it will keep lateral errors as small as possible, and at the same time, it will reduce the number of crash accidents.

#### 5.4.3.1/ SHARP CURVE ESTIMATION

Sharp curves are defined as dangerous curves if their radius are small or central angles are big [75]. This type of curves deserve special attention due to the fact that they correspond to locations where the vehicle needs to decrease speed in order to take the curve

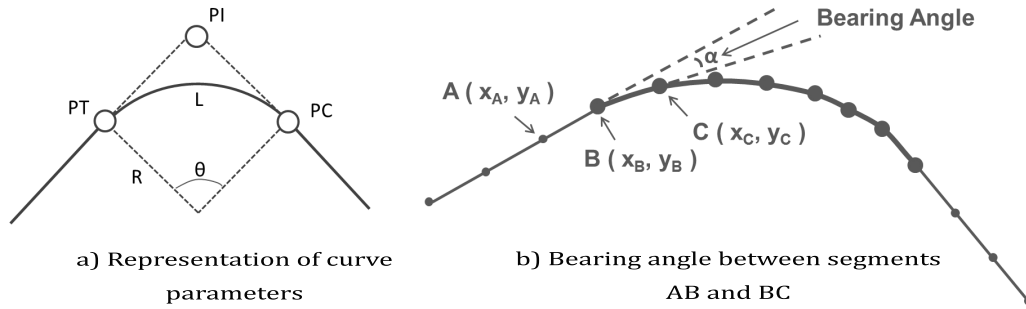


FIGURE 5.8 – Curve parameters representation

as if a human was driving, keeping small lateral errors [137].

In order to identify sharp curves, we first localized all possible curves in the traveled GPS path based on Li et al. [75] work. Localizing curves, means identifying their start and end points, which are also called points of tangency (PT) and points of curvature (PC) respectively. Once these points are identified, their curve characteristics can be computed : point of intersection (PI), curve length (L), radius (R) and central angle ( $\theta$ ). An illustration of all these characteristics is shown in Fig. 5.8a.

Classifying points as being part of a curve or of a straight segment, is the first step to identify start and end points (PT and PC) in a curve. This procedure is performed by computing the bearing angle ( $\alpha$ ) between two consecutive segments formed with 3 points (A,B,C) as seen in Fig. 5.8b, using Equation 5.5.

$$\alpha = \cos^{-1} \left( \frac{(x_B - x_A)(x_C - x_B) + (y_B - y_A)(y_C - y_B)}{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2} \times \sqrt{(x_C - x_B)^2 + (y_C - y_B)^2}} \right) \times \frac{180}{\pi} \quad (5.5)$$

Once the bearing angle is computed, a threshold value is assigned to determine if the point B is considered as part of a curve or not. If the angle  $\alpha$  is bigger than the defined threshold, the center point B is described as being part of a curve. Choosing the right threshold is a very important step, since curvature identification depends on this value. In this work a threshold value of  $1.25^\circ$  was chosen as proposed in [75] after evaluating curvature identification with different values. It is worth mentioning that values smaller than 1.25 gave us false positives, as almost every point was considered being part of a curve, and values bigger than 1.25 discarded long and smooth curves.

Based on Li et al. work [75], curves were classified as simple or compound. A compound curve is a curve formed with at least 2 consecutive curves separated by a certain distance between each other. In our datasets, this distance was set to 10.5 m which is the approximation of considering 3 consecutive points in our preprocessed GPS path. An example



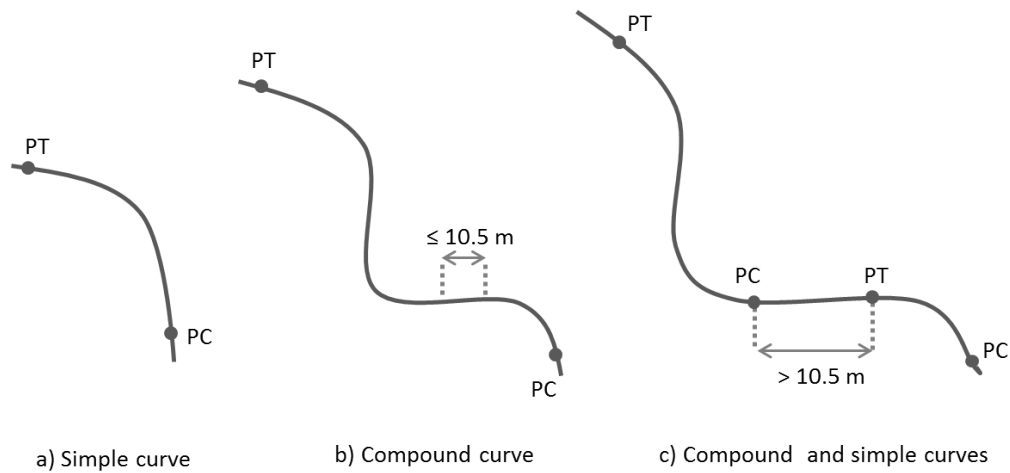


FIGURE 5.9 – Curve classification

of these 2 types of curves is illustrated in Fig. 5.9.

After each curve is defined by its start and end points (PT and PC), its geometric information, consisting of its radius (R), length (L) and central angle ( $\theta$ ), are computed through the Equations 5.6-5.8. In order to perform these computations, the central point of the curve (O) is identified following the work [75].

$$R = \sqrt{(x_{PC} - x_O)^2 + (y_{PC} - y_O)^2} \tag{5.6}$$

$$C = \sqrt{(x_{PT} - x_{PC})^2 + (y_{PT} - y_{PC})^2} \tag{5.7}$$

$$\theta = 2 \times \sin^{-1} \left( \frac{C}{2R} \right) \times \frac{180}{\pi} \tag{5.8}$$

Curve length (L) was estimated by summing up the euclidean distances between the segments that form the curve.

As we are interested in considering only dangerous curves, sharp curves were categorized if their angle ranges from  $30^\circ$  to  $180^\circ$ , or their radius is between 5 and 18 meters according to the American Association of State Highway and Transportation Officials' (AASHTO's) "A Policy on Geometric Design of Highways and Streets" [17]. Fig. 5.10 shows an example of the curve classification according to [75] and marked in red circles are the sharp curves identified with our method. In this work, only sharp curves are considered to adjust the speed in path tracking simulations.

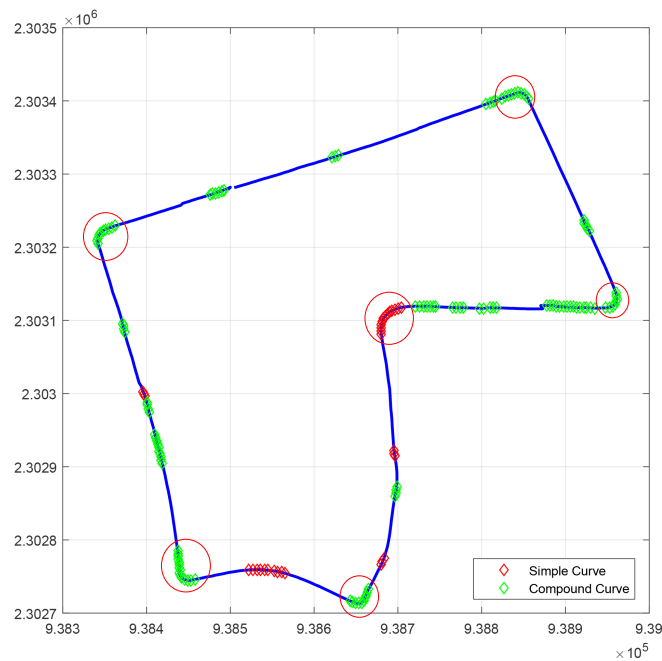


FIGURE 5.10 – Curve classification of UTBM-2 dataset. Sharp curves are marked with red circles.

#### 5.4.3.2/ SPEED COMPUTATION FOR SHARP CURVES

Curve speed estimation depends on centripetal and centrifugal forces. Centripetal force, which relies on the friction between tires and the roadway, is the one that makes the vehicle follow the curve while centrifugal force tries to move the vehicle outwards the curve. In order to compensate this last force, road curves are designed with an inclination known as banked angle ( $\theta$ ) or super-elevation angle ( $e$ ). In other words, ideal speed in curves will be computed depending on the curvature of the path ( $k$ ), friction coefficient ( $\mu$ ) and super-elevation angle ( $e$ ) as shown in figure 5.11, where  $N$  is the normal force,  $f$  is the friction force,  $\mu$  is the friction coefficient,  $m$  is the vehicle mass,  $g$  is the gravity and  $F_{net}$  is the net force.

Through the normal force ( $N$ ), friction force ( $f$ ) and weight force ( $mg$ ) vectors, the centripetal force is defined as :

$$F_C = \frac{mv^2}{r} \quad (5.9)$$

Where  $m$  represents vehicle mass,  $v$  vehicle speed and  $r$  the curve radius.

Summing up the vertical components, normal force can be described following Equation

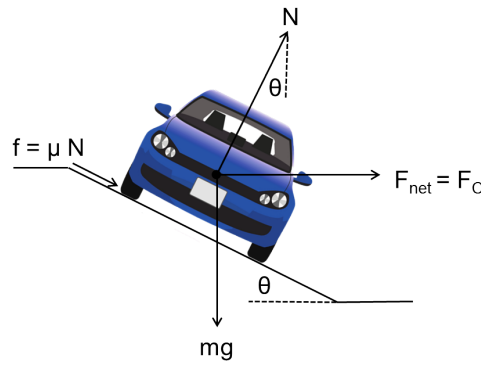


FIGURE 5.11 – Centripetal force diagram

5.10.

$$N \cos \theta = mg + f \sin \theta \quad (5.10)$$

$$N = \frac{mg}{\cos \theta - \mu \sin \theta} \quad (5.11)$$

The net force is calculated through the horizontal components :

$$F_{net} = N \sin \theta + N \mu \cos \theta \quad (5.12)$$

Substituting the normal force in Equation 5.12,

$$F_{net} = \left( \frac{mg}{\cos \theta - \mu \sin \theta} \right) (\sin \theta + \mu \cos \theta) = \frac{\tan \theta + \mu}{1 - \mu \tan \theta} mg \quad (5.13)$$

Since friction coefficient ( $\mu$ ) usually varies from 0.1 to 0.16 [124], the value in the denominator tends to be around one, so it can be discarded. Now, the net force can be defined as the centripetal force ( $F_{net} = F_{centripetal}$ ), obtaining the following equation :

$$(\tan \theta + \mu)g = \frac{v^2}{r} \quad (5.14)$$

Using super-elevation  $e = \tan \theta$  and curvature information  $r = \frac{1}{k}$ , we substitute both terms into 5.14. The ideal speed will be given by equation 5.15, which is exactly the same definition as in [124].

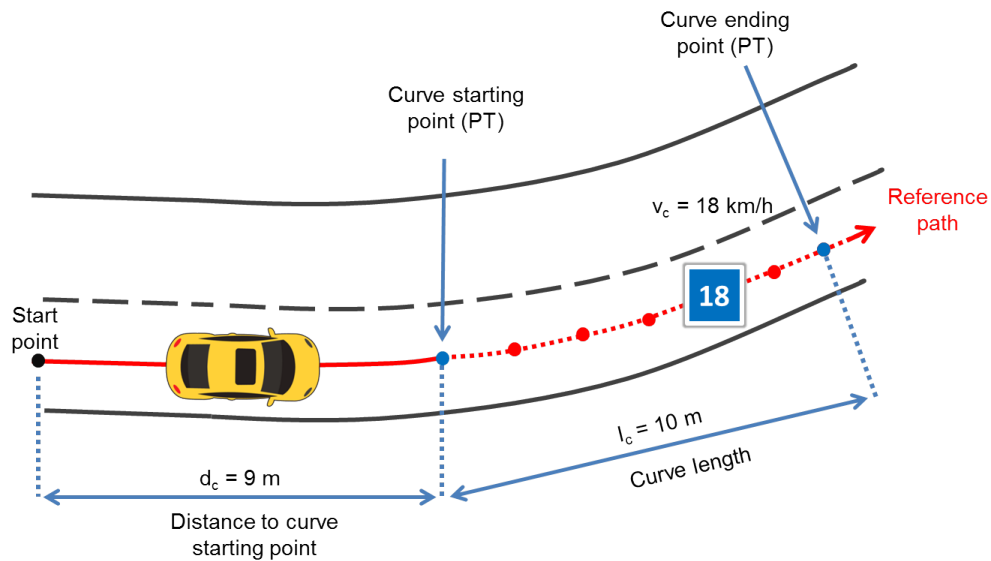


FIGURE 5.12 – Curve driving scenario.

$$v = \sqrt{\frac{(e + \mu)g}{k}} \quad (5.15)$$

Following this equation, convenient speeds for each sharp curve ( $v_c$ ) were estimated considering a super-elevation value between 6% and 12%, which, according to the American Association of State Highway and Transportation Officials' (AASHTO's) [17], is the value defined for rural and urban roads.  $k$  is the curvature information computed as the reciprocal of the radius,  $k = \frac{1}{r}$ .

Finally, given a sharp curve detected in the traveled GPS path, we computed the distance ( $d_c$ ) from the reference start point of the path to each curve starting point (PT) as shown in figure 5.12. This distance together with its respective curve length ( $l_c$ ) and the convenient speed ( $v_c$ ) are the parameters passed to the speed negotiation algorithm, detailed in the next subsection.

#### 5.4.4/ SPEED NEGOTIATION ALGORITHM

In typical manual driving situations, drivers notice the upcoming speed limit sign or curve, and slow down or accelerate (depending on the current driving speed) to handle the road properly. Automating this procedure in ground vehicles requires knowledge ahead to adjust vehicle's speed before entering a sharp curve or arriving to a new speed limit zone. Our work covered this knowledge extraction in subsections 5.4.2 and 5.4.3.

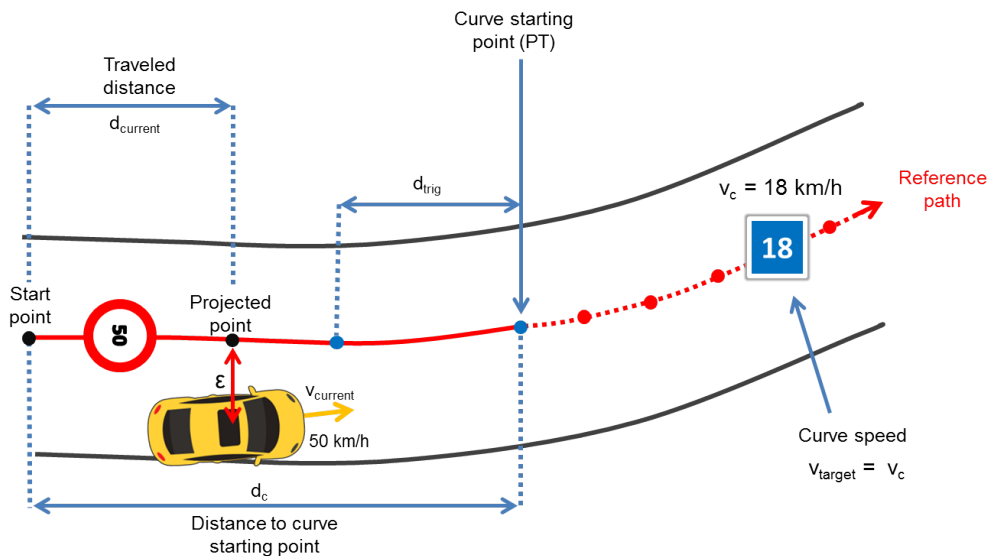


FIGURE 5.13 – Curve driving scenario projecting vehicle position into the reference path.

The speed negotiation algorithm tries to simulate human driving behavior based on the knowledge extracted from curves and speed limits to decide the ideal speed ( $v_{ideal}$ ) the vehicle should be traveling with. This negotiation depends mainly on the actual vehicle position in the GPS path, and the analysis of the input parameters coming from curve and speed limits as shown in Fig. 5.6.

Initially, the lateral control system computes the lateral error ( $\epsilon$ ) and the current vehicle traveled distance ( $d_{current}$ ) in the path (see Fig. 5.13). This computation is estimated by projecting the current vehicle position on the reference path through spline interpolation to match the GPS points.

Once the projected vehicle position is known, the speed negotiation algorithm analyses which type of road segment (curve or speed limits) is closer to the current vehicle position and computes a trigger distance. This trigger distance ( $d_{trig}$ ) has the objective to take into account a smooth deceleration behavior to reach the ideal speed with comfort. It is calculated with the following equation :

$$d_{trig} = \frac{v_{target}^2 - v_{current}^2}{2a} \quad (5.16)$$

Where :

- $v_{target}$  is either the speed limit ( $v_l$ ) or curve speed ( $v_c$ ),
- $v_{current}$  is the current vehicle's speed, and
- $a$  is the deceleration value to be applied.

**Data :**  $d_{current}, v_{current}, v_l, d_l, v_c, l_c, d_c$   
**Result :**  $v_{ideal}$   
 $aux\_d_c = d_c - d_{current};$   
 $aux\_d_l = d_l - d_{current};$   
**if**  $aux\_d_c < aux\_d_l$  **then**  
    |  $v_{ideal} = v_c;$   
**else**  
    |  $v_{ideal} = v_l;$   
**end**  
Compute  $d_{trig} = \frac{v_{ideal}^2 - v_{current}^2}{2a};$   
**if**  $v_{current} > v_{ideal}$  **then**  
    | **if**  $v_{ideal} == v_c$  **then**  
        | **if**  $(d_{current} \geq d_c - d_{trig})$  **AND**  $(d_{current} \leq d_c + l_c)$  **then**  
            | Apply  $a_{neg}$  until  $v_{current} = v_{ideal};$   
            **end**  
        | **else**  
            | **if**  $(d_{current} \geq d_l - d_{trig})$  **AND**  $(d_{current} < d_l)$  **then**  
                | Apply  $a_{neg}$  until  $v_{current} = v_{ideal};$   
                **end**  
            **end**  
        | **end**  
    | **else**  
        | Apply  $a_{max}$  until reaching  $v_{ideal};$   
    | **end**

**Algorithm 1 :** Speed negotiation pseudo-algorithm

An appropriate deceleration value is considered according to a study made by Maurya et al. [76] which compares several deceleration behaviors at different speeds in various vehicle types. In our case, we set a maximum acceleration ( $a_{max}$ ) and deceleration ( $a_{neg}$ ) to  $2 \text{ m/s}^2$  to provide a comfortable transition between speeds and avoid longitudinal jerks.

If sharp curved positions overlap with zones where a speed limit is lower than 50 km/h, our algorithm gives priority to the lower speed, which is usually the curve speed.

Considering all aspects discussed above, the speed negotiation algorithm works as follows :

At each time step, an ideal speed ( $v_{ideal}$ ) results from the speed negotiation algorithm. This speed is used by the lateral control system to calculate an appropriate angle ( $\theta$ ) to be applied by the vehicle steering wheel. During every control cycle, the lateral control system sends to the vehicle, speed and angle parameters to provide a smooth tracking performance.

The goal of reducing ahead the speed before arriving to a curve or a speed limit zone, will provide the passenger comfort avoiding abrupt rapid decelerations. At the same time,

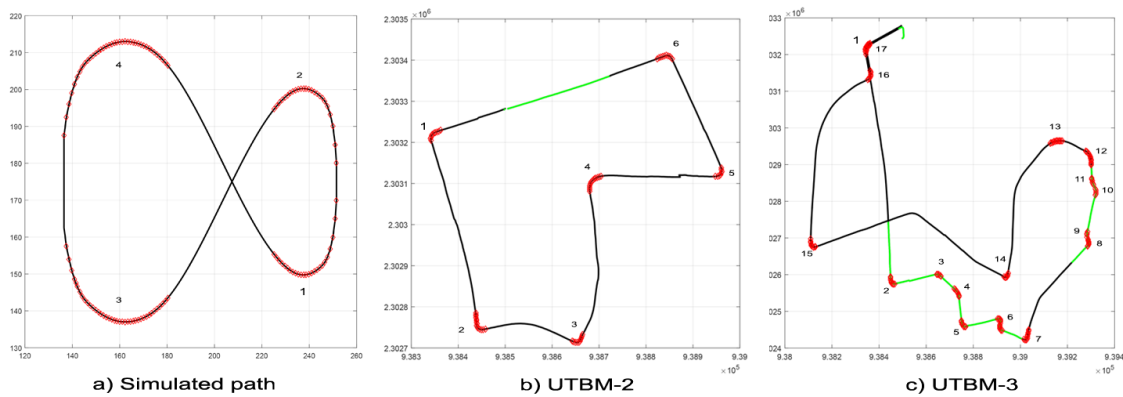


FIGURE 5.14 – Datasets with sharp curves and speed limits marked. Normal black line represents a speed limit of 50 km/h, green line a speed limit of 30 km/h and sharp curves are in red.

respecting the ideal speed definition will increase safety and provide more time to react in case hazard situations occur. As a result, lateral errors will decrease, which is the aim of path tracking algorithms.

## 5.5/ RESULTS

In this section we perform simulations with different lateral control algorithms on a simulated path and real datasets. This with the aim to show how tracking accuracy is improved if correct speed definitions are taken into account for each segment in the traveled path. Consequently, a sense of comfort and safety will be provided to the passenger reducing traffic accidents and severe injuries [66].

Three paths (one created and two real) were considered for the simulations. Each path has different length, curve characteristics and speed limits. The real datasets (traveled paths) were acquired by Qiao et al. [125] with the university equipped vehicle in Belfort, France. The data of the paths is based only on GPS information captured by GPS and RTK-GPS receivers. The simulated path was created with the aim to compare how noise affects the tracking performance of lateral control algorithms.

A visual representation of the considered paths can be seen in Fig. 5.14. They are briefly presented below.

1. **Simulated path.** This path is composed of 210 continuous points in a 8 shape form. Its total distance is about 374 m with 4 sharp curves identified. Since it is a virtual path, we set the speed limit for the entire track to 50 km/h as it is the default

speed limit in urban areas in France [12].

2. **Dataset UTBM-2.** The traveled path contains 541 points after the preprocessing step performed as described in subsection 5.4.1. The distance of this track is about 2.2 km with detected speed limits of 30 km/h in school zones. 6 sharp curves were identified with different geometric characteristics.
3. **Dataset UTBM-3.** This path is the longest one with a distance of about 4.5 km. It is represented by 1136 points after the preprocessing step. The number of sharp curves (dangerous curves) detected was 17. The speed limits for this path are 30 km/h in residential areas and 50 km/h otherwise.

Different path tracking simulations were performed through the tool developed by Lombard et al. [144]. These simulations include the analysis of different lateral control algorithms with and without considering the ideal speeds obtained by our DSA method.

In order to compare the performance of our proposed DSA method between the different steering control algorithms : 1) Pure Pursuit (PP), 2) Stanley, 3) Alice and 4) Lombard ; we compute the root mean square error ( $e_{rms}$ ) per each dataset for each of the four algorithms as follows :

$$e_{rms} = \sqrt{\frac{\sum (p_d(t) - p_c(t))^2}{m}} \quad (5.17)$$

where  $p_d$  and  $p_c$  represent the desired and current vehicle position respectively. The difference between  $p_d$  and  $p_c$  is the lateral error, while  $m$  is the total number of sampling results every 400ms (latency of the system). Then, the average error on the total number of datasets per each method is calculated as :

$$\overline{e_{rms}} = \frac{1}{N} \sum e_{rms} \quad (5.18)$$

Where  $N$  is the number of datasets per algorithm to be evaluated. In the case of curve evaluation,  $N$  equals to 3 (2 real and 1 simulated datasets) while for speed limits,  $N$  equals to 2 (the 2 real datasets).

In France, the speed limit in urban areas is defined as 50 km/h [12], which is the maximum speed assigned in the simulations. Other than that, in our datasets, speed limit zones of 30 km/h were identified and considered to adjust vehicle speed. Regarding curve speeds, they vary from 14 to 36 km/h depending on the sharpness detected for each curve.



TABLE 5.1 – Error comparison in zones of 30 km/h with different Lateral Control methods. Results are expressed in meters.

Dataset	Stanley	Stanley DSA	Alice	Alice DSA	PP	PP DSA	Lombard	Lombard DSA
UTBM-2	0.0182	0.0158	0.0452	0.0416	0.1005	0.0554	0.0845	0.0550
UTBM-3	0.0210	0.0112	0.2840	0.1934	0.3346	0.2823	0.3374	0.2817
Average	0.0196	0.0135	0.1646	0.1175	0.2175	0.1688	0.2109	0.1683

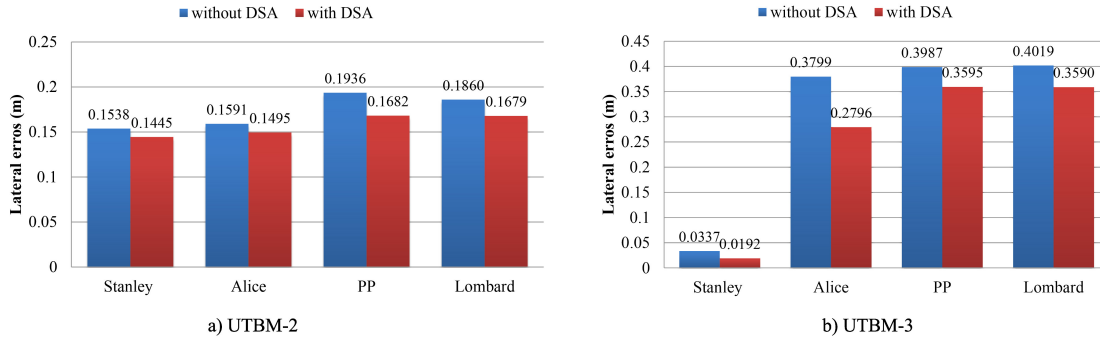


FIGURE 5.15 – Root mean square error ( $e_{rms}$ ) comparison of lateral errors obtained in 30 km/h speed limit segments.

We analyzed lateral errors produced when speed limits change from 50 to 30 km/h in UTBM-2 and UTBM-3 datasets (see Table 5.1). In other words, we analyzed lateral errors in segments of 30 km/h (green zones as seen in Fig. 5.14) and compared the results for each lateral control method. Speed limit zone of 30 km/h in UTBM-2 is a straight segment, and even if the lateral errors in all the methods are similar (15-20 cm error), the performance of all methods is improved considering our DSA method (see figure 5.15a). Lateral errors in UTBM-3 dataset vary more due to its road geometry, therefore, improvements with our DSA method are more noticeable as shown in figure 5.15b.

In average, the evaluation of speed limits in segments of 30 km/h improved all the methods with our DSA, but not more than 5 cm as seeing in figure 5.16. It is worth mentioning that the literature has proven that respecting speed limits increases safety [38, 23, 31, 37]. Autonomous vehicles capable of adjusting automatically speed limits will be preferred, since, in real scenarios, drivers go beyond the authorized speeds being prone to cause accidents.

Now, we will focus our analysis on the most dangerous segments, which are the sharp curves. Tables 5.2, 5.3 and 5.4 show lateral error results obtained by the different steering control algorithms in the detected curves for each path.

We will start comparing the simulations performed between paths which contain or not

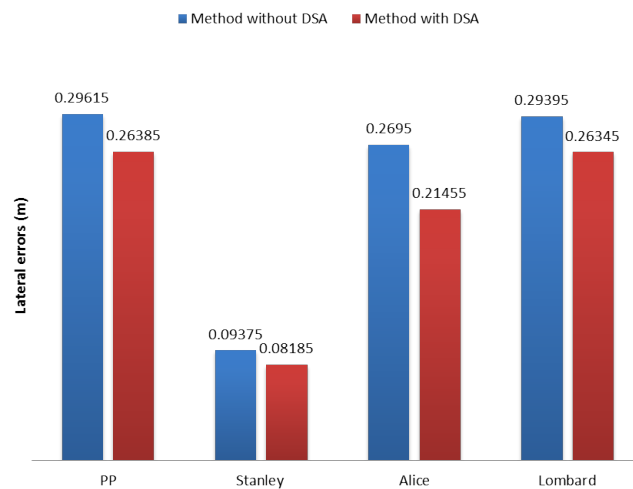


FIGURE 5.16 – Average root mean square error ( $\overline{e_{rms}}$ ) of lateral error comparison between methods for 30 km/h speed limit segments.

TABLE 5.2 – Error comparison in sharp curves of different Lateral Control methods for the Simulated path. Results are expressed in meters.

Curve	Stanley	Stanley DSA	Alice	Alice DSA	PP	PP DSA	Lombard	Lombard DSA
1	0.0599	0.0388	0.7637	0.6249	0.1935	0.2257	0.1854	0.0613
2	0.0762	0.0302	0.8286	0.6128	0.7972	0.0603	0.1752	0.04807
3	0.1029	0.0508	1.2195	0.9323	0.4351	0.2239	0.2223	0.09659
4	0.0493	0.0315	0.5757	0.4726	0.2107	0.1204	0.1531	0.04335
Av	0.0721	0.0378	0.8469	0.6607	0.4091	0.1576	0.1840	0.06233

noisy information. Regarding tracking error results obtained in the simulated path (Table 5.2), we can see that most of lateral errors are the smallest (except for Alice method) compared to errors obtained in real datasets (Tables 5.3 and 5.4). A visual representation of this comparison can be seen in Fig. 5.17a. We can confirm that noise is an important factor for tracking algorithms and it deserves special attention to reduce it before controlling autonomous cars. In the literature, authors have combined different information to deal with noise, e.g. taking into account the dynamics of the car (position, orientation, speed) [50].

The improvements of each method, with and without considering speed regulations in curves, range from 22% to 88% depending on the path. For example, in the simulated path (see Fig. 5.17b), tracking error is reduced by 19 cm in Alice method (22% improvement). In UTBM-2, the method with the largest root mean square errors is Pure Pursuit (1.2 m), for which the error is reduced by 72 cm with our DSA (see Fig. 5.17c). A visual representation of lateral errors on UTBM-2 dataset is shown in Fig. 5.18, which illustrates

TABLE 5.3 – Error comparison in sharp curves of different Lateral Control methods for UTBM-2 dataset. Results are expressed in meters.

Curve	Stanley	Stanley DSA	Alice	Alice DSA	PP	PP DSA	Lombard	Lombard DSA
1	0.0722	0.0288	0.5747	0.3308	0.8657	0.2451	0.4231	0.0552
2	0.0743	0.0245	0.6572	0.4945	0.7135	0.2372	0.5050	0.0589
3	0.2586	0.1797	1.2132	0.7815	1.5847	0.6940	0.9028	0.1292
4	0.0574	0.0265	0.5724	0.2591	0.5528	0.2396	0.2616	0.0382
5	0.1214	0.0449	1.0485	0.5650	1.0909	0.3627	0.5347	0.0751
6	0.1001	0.0441	0.9109	0.4528	1.4346	0.7075	0.7563	0.0687
Av	0.114	0.0581	0.8295	0.4807	1.0404	0.4143	0.5639	0.0709

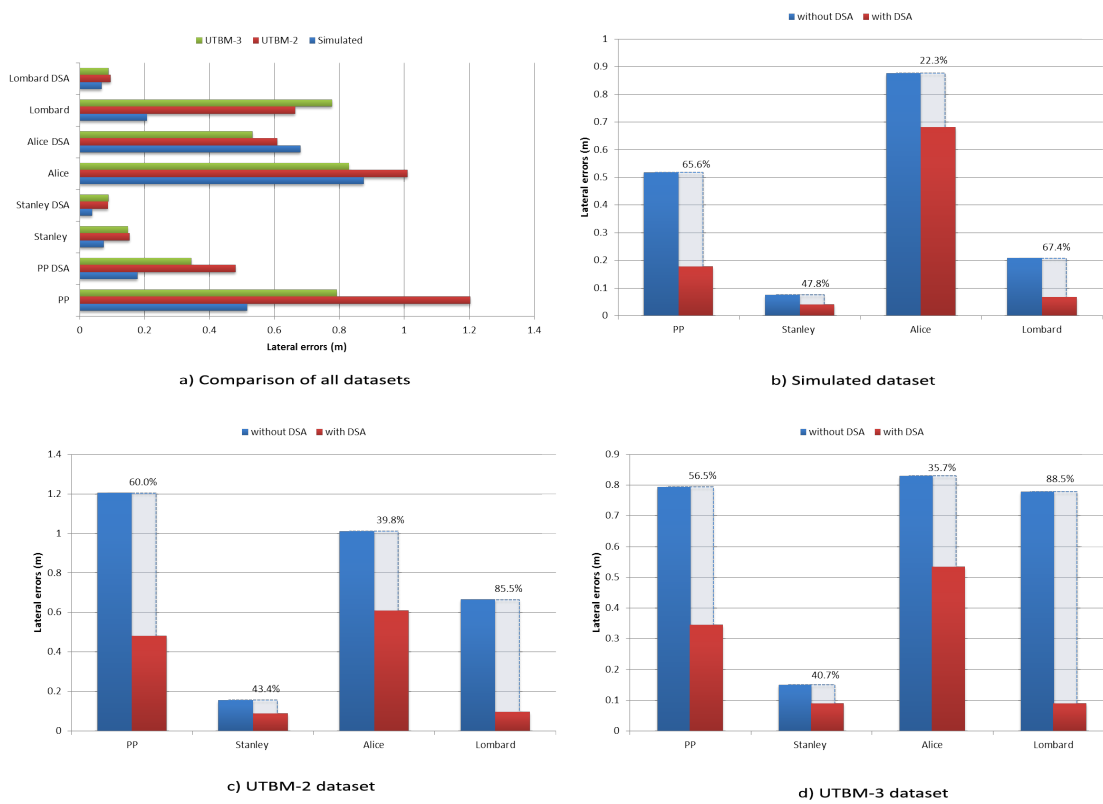


FIGURE 5.17 – Root mean square error ( $e_{rms}$ ) comparison of lateral errors obtained in sharp curves for each dataset.

how our DSA method is able to reduce errors significantly in sharp curves. At the same time, it is clearly noticeable that the biggest lateral errors are present in sharp curves confirming the need to adapt vehicle' speed. Furthermore, in UTBM-3 dataset (see Fig. 5.17d), tracking error is reduced by about 45 cm in the Pure Pursuit method, while for Lombard method, it is reduced by 69 cm (88% improvement).

In general, the performance of our DSA method in sharp curves showed significant (at least 10 cm error) improvements for most of the methods (see Fig. 5.19), except for

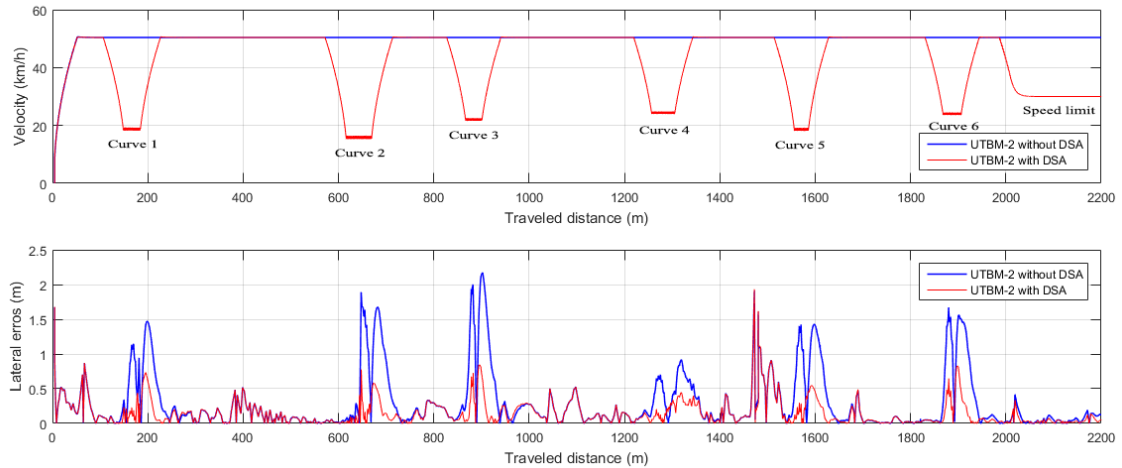


FIGURE 5.18 – Speed and lateral error comparison in UTBM-2 dataset using PP method with and without DSA.

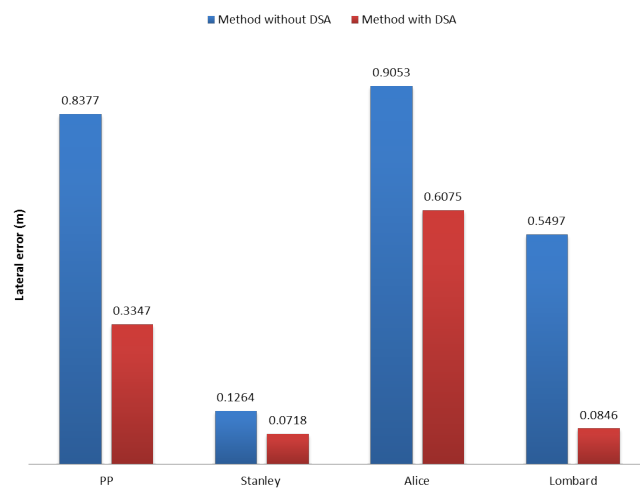


FIGURE 5.19 – Average root mean square error ( $\overline{e_{rms}}$ ) of lateral error comparison between methods for sharp curves.

Stanley method (improvement of around 5 cm). The reason behind this behavior is the definition of its goal point. Since Stanley method is based on Pure Pursuit (PP), and the goal point is defined with a short distance, it does not suffer a lot from the "cutting corners" effect.

As tracking errors for speed limits decrease, but not significantly (1 cm for Stanley, 5 cm for Alice and 3 cm for Pure Pursuit and Lombard), we will base our final conclusion on the results obtained in sharp curves. Tracking errors with our DSA method improved significantly 3 methods, Pure Pursuit (PP), Alice and Lombard. Even though the improvements for Stanley method are not considered significant, there is an improvement of about 5 cm when speed adaptation is used. The 2 methods which benefit the most with DSA are

Pure Pursuit and Lombard (as seen in Fig. 5.19) with about 50 and 47 cm difference respectively for sharp curves. For Alice method, the difference is about 30 cm which is also a very good improvement. This proves that, having the adapted speeds to pass through out sharp curves, certainly leads to provide the passenger confidence and safety for taking a ride in an autonomous vehicle.

One of the main limitations for our proposed method, resides in GPS noise. Nevertheless, this limitation can be treated combining more information, for example vehicle dynamics [50] to analyze position, orientation and speed. Inaccurate curvature geometry extraction is another limitation when noisy GPS information is present. Nonetheless, this last problem can be overcome extracting curve parameters from digital maps.

## 5.6/ CONCLUSION AND FUTURE WORKS

In this chapter, we have proposed a real time Dynamic Speed Adaptation (DSA) method based on the analysis of speed limits and curvature information. A speed limit database of the traveled paths was created and considered as input for our DSA. Curvature information extraction allowed DSA to identify sharp curves and to compute the recommended speed for each one. The speed limit database and curve geometric information are extracted off-line from GPS paths in order to provide knowledge ahead of the road while traveling on it.

The tracking performance using our approach, showed significant reduction of lateral errors. This improvement may potentially prevent accidents and reduce severe injuries in real driving scenarios, which is exactly the aim of any autonomous vehicle.

The main advantage of our proposed DSA method is its adaptability, since it can be implemented in the vehicle lateral control system independently of the steering wheel angle computation. Besides, it enables the car to drive with recommended speeds through straight and curved segments respecting traffic regulations. Also, it is smart enough to localize upcoming sharp curves and apply a comfortable deceleration value before arriving to them, reaching their ideal speeds to traverse them.

Future works will include the collection and analysis of vehicle dynamics in order to deal with GPS noise. This information will correct noisy points in the path by comparing the vehicle's speed and orientation with the GPS data.

Furthermore, we will incorporate the visual information extracted in Chapter 4 in order to

bring perception capabilities to the autonomous vehicle. This information will allow us to : (1) detect the road if GPS is not available ; (2) identify speed limit traffic signs in case no speed limit database is accessible ; (3) recognize lane markings on the road to guide the algorithm for road boundary detection and shape estimation, (4) detect pedestrians and vehicles to take them into account for obstacle avoidance, (5) locate crosswalks in the road to pay attention for possible braking situations if pedestrians are positioned close to them or overlapping (pedestrian is already crossing using the crosswalk area). Anyhow, we know that visual perception is necessary for the autonomous vehicle due to its utility for the different applications to provide safety. For this reason, the environment perception module plays an important role to achieve full automation (level 5), which is the aim of every research group and private company involved in this field.

TABLE 5.4 – Error comparison in sharp curves of different Lateral Control methods for UTBM-3 dataset. Results are expressed in meters.

Curve	Stanley	Stanley DSA	Alice	Alice DSA	PP	PP DSA	Lombard	Lombard DSA
1	0.0763	0.0328	0.6984	0.3589	0.5012	0.2007	0.4840	0.0574
2	0.0910	0.0311	0.7711	0.4342	0.5934	0.1141	0.5959	0.0530
3	0.0545	0.0258	0.3409	0.2619	0.4462	0.1409	0.4498	0.0337
4	0.0421	0.0196	0.3409	0.1912	0.4056	0.0975	0.3857	0.0218
5	0.0893	0.0296	0.7401	0.4422	0.6162	0.0948	0.6225	0.0519
6	0.4570	0.4209	1.4465	1.0717	1.3936	1.3865	1.4099	0.2663
7	0.1697	0.1054	1.0596	0.6294	0.8554	0.565	0.8769	0.0835
8	0.1781	0.0705	1.1031	0.7269	0.9603	0.3007	0.9640	0.0818
9	0.0718	0.0297	0.3765	0.3331	1.1918	0.3802	1.2084	0.0523
10	0.0753	0.0278	0.5634	0.3694	1.0366	0.1566	1.0426	0.0488
11	0.0344	0.0183	0.2441	0.1671	0.5754	0.4024	0.5825	0.0240
12	0.0472	0.0247	0.4376	0.2123	0.2039	0.1248	0.2053	0.0306
13	0.0209	0.0193	0.2167	0.0885	0.1104	0.1312	0.1101	0.0145
14	0.1573	0.0669	0.9394	0.6268	0.9873	0.1431	0.9973	0.0613
15	0.1027	0.0389	0.9034	0.4698	0.5558	0.2250	0.5589	0.0651
16	0.1171	0.0424	0.7964	0.4465	0.9738	0.3490	0.8710	0.0706
17	0.1505	0.0615	0.5531	0.6594	0.934	0.2574	0.8212	0.0813
Av	0.1138	0.0627	0.6783	0.4405	0.7259	0.2982	0.7168	0.0646



## CONCLUSIONS AND FUTURE WORKS





# CONCLUSIONS AND FUTURE WORKS

This chapter summarizes the contributions of the thesis together with its main conclusions and recommendations for future research. Detailed conclusions for each individual topic can be found in the corresponding chapters.

## 6.1/ CONCLUSIONS

We have addressed two out of the five components for intelligent autonomous vehicles : the environment perception and vehicle control. A reliable environment perception is probably one of the most important parts for automated vehicles because it enables the vehicle understand what is happening in its surroundings. Analyzing the visual information was performed through the use of deep learning techniques, which, after a careful review of the state of the art, led to our proposed methods presented in chapters 3 and 4.

The global localization of the vehicle was carried out through an RTK GPS for accurate positioning. This information was used for the vehicle control to track the recorded path applying lateral and longitudinal movements. A comparative analysis of different control algorithms was performed in chapter 5 leading to our contribution for dynamic speed adaptation when curves are perceived.

A general conclusion for each of our contributions is presented in the following paragraphs regarding each chapter.

Our **first contribution** relates to data information. Three datasets were introduced in chapter 2 with the aim to contribute in some research areas with new challenges. The first dataset, European Traffic Sign Dataset (ETSD), was proposed as a standard definition for European traffic signs with 164 classes. This dataset gathers information from 6

countries and deals with intraclass variability. We are confident this dataset will help the European community to achieve the connected and automated mobility plan expected by 2030 [177]. The second dataset, UTBM-2, was labeled in a pixel manner with 27 semantic classes and 10 instance ones. This dataset provides visual information in French urban scenarios and includes the GPS position for each image. At the same time, the instances for each traffic sign are labeled with its corresponding traffic sign class following the definition of our proposed ETSD. Lastly, the extended GTSDb dataset, labeled for detection, semantic and instance segmentation approaches, contains labels for only traffic signs. The difference between the original and the proposed extended version relies on the number of traffic signs labeled and consequently the number of classes considered. We provide three different formats of ground truth, (1) a txt file with the Rols position and label (detection approaches), (2) an image labeled in a dense pixel manner with the traffic sign class (semantic segmentation) and (3) an image labeled in a dense pixel manner with the traffic sign class and the instance IDs for each sign (instance segmentation).

Regarding our **second contribution** presented in chapter 3, we have proposed a CNN architecture to perform traffic sign classification. Our classifier called Class\_CNN, inspired by VGG-16 network, is designed with 5 blocks of convolutional layers, a fully connected layer and a Softmax classifier. The CNN uses a new regularization technique (Dropblock) which made the performance improved while at the same time, the learning process became more stable. A comparison study between different CNN architectures is provided with evaluations on a common standard dataset (GTSRB) and our proposed ETSD dataset.

In our **third contribution** (chapter 4), we have proposed a system for visual environment perception. The pipeline of our system is composed of two modules, one in charge of segmenting each image pixel in 15 classes of interest, and the other one to detect and classify individual traffic signs. The traffic sign recognition first detects traffic sign instances through Mask R-CNN and, using the masks, we extract the image region, pre-process it and pass it to either our proposed category classifier (Cat\_CNN) or to the class classifier (Class\_CNN). Several filtering processes are taken into account to discard any background information and provide a reliable output. Each module is evaluated separately to account for segmentation results, traffic sign detection and traffic sign classification. The experimental results for the semantic segmentation module showed that, identifying the borders between classes is a challenging task no matter the different methods' strategies to include context and spatial information. As the classes imbalance will always be a problem for learning approaches, data augmentation techniques should always be ap-

plied. Regarding the results for traffic sign recognition, we have proved that, for traffic sign classification, a deeper CNN (Class\_CNN) performs better than a simpler one (Cat\_CNN) and that with the inclusion more challenging signs, like text based or with very different aspect ratios, the system performance decreases. Nevertheless, its inclusion is necessary for a correct interpretation of the traffic driving rules in the perceived environment.

For our **fourth and last contribution**, a method to improve vehicle control was introduced in chapter 5. The proposed method is characterized for its adaptability as it can be incorporated for any vehicle control algorithm affecting only the speed computation. It requires an off line process to analyze the GPS recorded path which is used to find the sharp curved segments. In parallel, the identification of speed limits is required to create a speed limits database. When this information is extracted and passed to the vehicle control module, a speed negotiation algorithm evaluates the current vehicle position and decides the ideal speed to execute. If a sharp curve is detected to come, it computes the recommended speed to traverse it together with the distance ahead needed for the vehicle to apply constant deceleration. The speed limits serve as the maximum threshold the vehicle can travel with, even if the recommended curve speed is above the speed limit, the negotiation algorithm will not exceed it. Experimental results showed that our proposed method reduces the lateral errors in all the tested control algorithms, making it a reliable choice to improve path tracking.

## 6.2/ FUTURE WORKS

Based on the results obtained in this thesis, we believe that the following ideas will conduct to improvements.

The research community have already demonstrated that the engineering part to achieve automated/autonomous/intelligent vehicles is not the only factor needed it. It will require infrastructure, legal liability and social acceptance [177]. The European Union is already taking care of this transition phase and, in order to contribute a bit, we plan to include in our proposed ETSD, traffic signs from more European countries. The ideal solution for autonomous vehicles will be to recognize the same traffic sign (intraclass variability) no matter the European country where the vehicle will be located. At the moment, the dataset includes traffic signs from 6 countries but, we have found other public datasets captured in Italy [154] and Spain [51] that will help for this contribution.

Regarding the proposed visual environment perception system, currently it uses 3 sepa-

rate networks, one for semantic segmentation, one for instance segmentation and one for traffic sign classification. The system performance can be improved if the three models share a common feature extractor. This will involve a new CNN architecture definition with shared convolutional layers to define a common feature map. Similar to the Mask R-CNN flow, several branches and losses will be needed to perform every task of each CNN used in our system. In this way, the system execution performance could be improved. Nevertheless, a new limitation will be introduced as the traffic sign CNN classifier in this work (Class\_CNN) learned from a separate dataset (ETSD), which contains a wide variability and amount of samples in each class, and using a common feature map representing a whole scene will limit the classifier to learn only the traffic signs labeled in that dataset. Further research will be needed to solve this last problem or more data will be required to learn properly traffic sign classes (labeled in a pixel precision manner/masks in the scene image).

Another possible perspective for the environment perception system is to take advantage of depth information. Computing disparity maps from the stereo vision Bumblebee camera, depth estimation of certain classes of interest can be obtained for further use. For example, computing the distance to certain objects like cars or pedestrians, will complement the perception to help the vehicle control negotiate speeds if objects are placed too close to its actual position. At the same time, this distance information can be exploited to map local visual information into a global map like Openstreetmaps. Using the GPS coordinates together with the segmented image information and the depth map, we can select stationary key objects in the scene like traffic signs, traffic lights or crosswalks to create new raster layers in Openstreetmaps and map this new information if it is not already contemplated. A conversion from camera to global coordinate system will be required to make the necessary transformations and map correctly the information. This high level application could result useful for autonomous vehicles if, for example, speed limit information is not available in the map, we can add it or update it if changes have been made in the road.

Furthermore, in order to have a robust environment perception system, the fusion of multiple sensory data is required. Camera sensors are vulnerable to illumination changes and weather conditions but provide rich information about color and shape of the objects. Radar sensors provide accurate distance information but detect poorly the objects' shape. LIDARs, on the other hand, provide good shape and distance information but are expensive sensors. For all these previous reason, there have been approaches in the literature that make use of sensor fusion [26, 28, 30, 46] to cover all surroundings of the

vehicle. However, implementing this approach will require multiple sensor calibration to relate each sensor output in a common space. Once performed, the perception system will be capable to understand more precisely the environment overcoming the limitations of each sensor.

For the path tracking approach, fusing the perceived visual information would lead to improvements and, in the author's point of view, the following work implementation could be done :

- Removal of speed limit database creation. Thanks to the traffic signs recognition performed in the environment perception system, the vehicle control won't need to extract the speed limit information off-line, but will require the speed negotiation algorithm to apply immediately, after a speed limit sign (e.g. Zone 30) is recognized, deceleration/acceleration to reach the limit detected considering a maximum distance threshold. The amount of acceleration/deceleration will have to be computed according to the threshold defined and always considering a comfortable amount to avoid abrupt changes. In this way, the vehicle control will have to act as humans do after seeing a sign, react and apply the proper actions to obey the traffic speed limits authorized. Here, a priori map data could also be considered to double check the speed limit before the vehicle acts on it. Additionally to this last idea, we will have to implement a system verification to make sure that the information provided by the perception system is accurate and corresponds to the one provided by the map. Otherwise, if the perceived information is correct and missing in the map, the vehicle should contemplate only the perceived one. For this reason map verification is a relevant topic for autonomous vehicles and requires updated information to plan or follow precisely a path.
- As the our dynamic speed negotiation algorithm for the vehicle control module considers as well the road geometry, the segmented lane markings and road visual information could be exploited. Implementing on top of the segmentation, a system like RALPH : rapidly adapting lateral position handler [4], the road curvature could be computed through images together with the lateral offset of the vehicle relative to the lane center. The segmented road will have to be projected in a birds eye view choosing a trapezoid and sampling the image into a rectangular one in black and white. Following the hypothesis of road curvatures, the image is unwrapped with different distances to transform the lane markings into linear ones and then assign a respective score. The unwrapped image with the highest score will correspond to the best road curvature. We refer the reader to [4] for a more comprehensive

- overview. As a result, the off-line curvature analysis of the reference path could be avoided for path tracking. The only limitation of this approach is that lane markings are necessary and in urban environments this information is not always available.
- Alternatively, end-to-end approaches could also be considered to perform path tracking. As a future work plan, we have recorded several circuits around university campus where the stereo vision system together with GPS data and vehicle information (speed and steering wheel angle) were saved synchronizing each sensor to relate images with the others sensor's data. Hence, a model like the one proposed by NVIDIA [133] could be used to learn how to steer depending on the input image. Another approach could also be considered with the use of vehicle information. Image classification could be carried out relating the steering wheel angle captured at each image to distinguish the road into straight, curve left, curve right, sharp curve left or sharp curve right segments. An analysis and definition of steering wheel angle thresholds will be required. In this way, a CNN could perform road type classification and alleviate the GPS road geometry analysis performed off-line in our path tracking proposal. The only inconvenient of this last approach will be the speed computation as not every curve or sharp curve require the same speed to traverse them safely. Furthermore, the captured vehicle speed will have to be considered as well during training to output not only the road type segment but also the ideal vehicle speed.

With all the presented possible improvements, we believe that a path tracking approach with level 3 (conditional automation) could be feasible to achieve. We hope the work of this thesis could be helpful as a base for upcoming Ph.D students at UTBM that will conduct research in this area.

# BIBLIOGRAPHIE

- [1] FOR EUROPE-INLAND TRANSPORT COMMITTEE, E. C., AND OTHERS. **Convention on road signs and signals**. *United Nations Treaty Series 1091* (1968), 3.
- [2] FENTON, R. E. **Automatic vehicle guidance and control—a state of the art survey**. *IEEE Transactions on Vehicular Technology* 19, 1 (1970), 153–161.
- [3] NEUMAN, T. R., GLENNON, J. C., AND SAAG, J. B. **Accident analyses for highway curves**. No. 923. 1983.
- [4] POMERLEAU, D. **Ralph : Rapidly adapting lateral position handler**. In *Proceedings of the Intelligent Vehicles' 95. Symposium* (1995), IEEE, pp. 506–511.
- [5] SHLADOVER, S. E. **Review of the state of development of advanced vehicle control systems (avcs)**. *Vehicle System Dynamics* 24, 6-7 (1995), 551–595.
- [6] WELCH, G., AND BISHOP, G. **An introduction to the kalman filter**.
- [7] PICCIOLI, G., DE MICHELI, E., PARODI, P., AND CAMPANI, M. **Robust method for road sign detection and recognition**. *Image and Vision Computing* 14, 3 (1996), 209–223.
- [8] CHOWDHURY, M. A., WARREN, D. L., BISSELL, H., AND TAORI, S. **Are the criteria for setting advisory speeds on curves still relevant?** *Institute of Transportation Engineers. ITE Journal* 68, 2 (1998), 32.
- [9] LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. **Gradient-based learning applied to document recognition**. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324.
- [10] GAVRILA, D. M. **Traffic sign recognition revisited**. In *Mustererkennung 1999*. Springer, 1999, pp. 86–93.
- [11] ZHANG, Z. **A flexible new technique for camera calibration**. *IEEE Transactions on pattern analysis and machine intelligence* 22 (2000).
- [12] **French government, code de la route. article r413-1, r413-2, r413-3, r413-4**. Available online : <https://www.legifrance.gouv.fr>, 2001. (accessed on 04 April 2017).
- [13] CLERET, J., CHAUVIN, P., BERNARD, G., DUPRE, G., AND FLORIS, O. **Méthode de sélection des virages à signaler et niveau de signalisation à implanter**. *Guide pratique, Conseil Général de la Seine-Maritime* (2001).
- [14] DICKMANN, E. D. **The development of machine vision for road vehicles in the last decade**. In *Intelligent Vehicle Symposium, 2002. IEEE* (2002), vol. 1, IEEE, pp. 268–281.
- [15] VARHELYI, A. **Dynamic speed adaptation in adverse conditions : A system proposal**. *IATSS research* 26, 2 (2002), 52–59.
- [16] GRIGORESCU, C., AND PETKOV, N. **Distance sets for shape filters and shape recognition**. *IEEE transactions on image processing* 12, 10 (2003), 1274–1286.



- [17] AASHTO. **A policy on geometric design of highways and streets**, 5th ed. American Association of State Highway and Transportation Officials, Washington, DC, 2004.
- [18] BARNES, N., AND ZELINSKY, A. **Real-time radial symmetry for speed sign detection**. In *Intelligent Vehicles Symposium, 2004 IEEE* (2004), IEEE, pp. 566–571.
- [19] DE LA ESCALERA, A., ARMINGOL, J. M., PASTOR, J. M., AND RODRÍGUEZ, F. J. **Visual sign information extraction and identification by deformable models for intelligent vehicles**. *IEEE transactions on intelligent transportation systems* 5, 2 (2004), 57–68.
- [20] LOY, G., AND BARNES, N. **Fast shape-based road sign detection for a driver assistance system**. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on* (2004), vol. 1, IEEE, pp. 70–75.
- [21] MATAS, J., CHUM, O., URBAN, M., AND PAJDLA, T. **Robust wide-baseline stereo from maximally stable extremal regions**. *Image and vision computing* 22, 10 (2004), 761–767.
- [22] BAHLMANN, C., ZHU, Y., RAMESH, V., PELLKOFER, M., AND KOEHLER, T. **A system for traffic sign detection, tracking, and recognition using color, shape, and motion information**. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE* (2005), IEEE, pp. 255–260.
- [23] CARSTEN, O. M., AND TATE, F. **Intelligent speed adaptation : accident savings and cost–benefit analysis**. *Accident Analysis & Prevention* 37, 3 (2005), 407–416.
- [24] FLEYEH, H., AND DOUGHERTY, M. **Road and traffic sign detection and recognition**. In *Proceedings of the 16th Mini-EURO Conference and 10th Meeting of EWGT* (2005), pp. 644–653.
- [25] BARNES, N., AND LOY, G. **Real-time regular polygonal sign detection**. In *Field and Service Robotics* (2006), Springer, pp. 55–66.
- [26] BRAID, D., BROGGI, A., AND SCHMIEDEL, G. **The terramax autonomous vehicle concludes the 2005 darpa grand challenge**. In *Intelligent Vehicles Symposium, 2006 IEEE* (2006), IEEE, pp. 534–539.
- [27] GAO, X. W., PODLADCHIKOVA, L., SHAPOSHNIKOV, D., HONG, K., AND SHEVTSOVA, N. **Recognition of traffic signs based on their colour and shape features extracted using human vision models**. *Journal of Visual Communication and Image Representation* 17, 4 (2006), 675–685.
- [28] THRUN, S., MONTEMERLO, M., DAHLKAMP, H., STAVENS, D., ARON, A., DIEBEL, J., FONG, P., GALE, J., HALPENNY, M., HOFFMANN, G., AND OTHERS. **Stanley : The robot that won the darpa grand challenge**. *Journal of field Robotics* 23, 9 (2006), 661–692.
- [29] WALKER, G. H., STANTON, N. A., AND YOUNG, M. S. **The ironies of vehicle feedback in car design**. *Ergonomics* 49, 2 (2006), 161–179.
- [30] BUEHLER, M., IAGNEMMA, K., AND SINGH, S. **The 2005 DARPA grand challenge : the great robot race**, vol. 36. Springer, 2007.
- [31] DRISCOLL, R., PAGE, Y., LASSARRE, S., AND EHRLICH, J. **Lavia—an evaluation of the potential safety benefits of the french intelligent speed adaptation project**. In *Annual Proceedings/Association for the Advancement of Automotive Medicine* (2007), vol. 51, Association for the Advancement of Automotive Medicine, p. 485.

- [32] EIDEHALL, A., POHL, J., AND GUSTAFSSON, F. **Joint road geometry estimation and vehicle tracking**. *Control Engineering Practice* 15, 12 (2007), 1484–1494.
- [33] GLASER, S., NOUVELIERE, L., AND LUSETTI, B. **Speed limitation based on an advanced curve warning system**. In *Intelligent Vehicles Symposium, 2007 IEEE* (2007), IEEE, pp. 686–691.
- [34] KUO, W.-J., AND LIN, C.-C. **Two-stage road sign detection and recognition**. In *Multimedia and Expo, 2007 IEEE International Conference on* (2007), IEEE, pp. 1427–1430.
- [35] LEIBE, B., CORNELIS, N., CORNELIS, K., AND VAN GOOL, L. **Dynamic 3d scene analysis from a moving vehicle**. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on* (2007), IEEE, pp. 1–8.
- [36] MALDONADO-BASCÓN, S., LAFUENTE-ARROYO, S., GIL-JIMENEZ, P., GÓMEZ-MORENO, H., AND LÓPEZ-FERRERAS, F. **Road-sign detection and recognition based on support vector machines**. *IEEE transactions on intelligent transportation systems* 8, 2 (2007), 264–278.
- [37] VLASSENROOT, S., BROEKX, S., DE MOL, J., PANIS, L. I., BRIJS, T., AND WETS, G. **Driving with intelligent speed adaptation : Final results of the belgian isatrial**. *Transportation Research Part A : Policy and Practice* 41, 3 (2007), 267–279.
- [38] AGERHOLM, N., WAAGEPETERSEN, R., TRADISAUSKAS, N., HARMS, L., AND LAHRMANN, H. **Preliminary results from the danish intelligent speed adaptation project pay as you speed**. *IET Intelligent Transport Systems* 2, 2 (2008), 143–153.
- [39] HAKLAY, M., AND WEBER, P. **Openstreetmap : User-generated street maps**. *IEEE Pervasive Computing* 7, 4 (2008), 12–18.
- [40] LINDEROTH, M., SOLTESZ, K., AND MURRAY, R. M. **Nonlinear lateral control strategy for nonholonomic vehicles**. In *American Control Conference, 2008* (2008), IEEE, pp. 3219–3224.
- [41] LUNDQUIST, C., AND SCHON, T. B. **Road geometry estimation and vehicle tracking using a single track model**. In *Intelligent Vehicles Symposium, 2008 IEEE* (2008), IEEE, pp. 144–149.
- [42] TAKASU, T., AND YASUDA, A. **Evaluation of rtk-gps performance with low-cost single-frequency gps receivers**. In *Proceedings of international symposium on GPS/GNSS* (2008), pp. 852–861.
- [43] WANG, C., HU, Z., AND UCHIMURA, K. **Precise curvature estimation by cooperating with digital road map**. In *Intelligent Vehicles Symposium, 2008 IEEE* (2008), IEEE, pp. 859–864.
- [44] BONNESON, J., PRATT, M., AND MILES, J. **Evaluation of alternative procedures for setting curve advisory speed**. *Transportation Research Record : Journal of the Transportation Research Board*, 2122 (2009), 9–16.
- [45] BROSTOW, G. J., FAUQUEUR, J., AND CIPOLLA, R. **Semantic object classes in video : A high-definition ground truth database**. *Pattern Recognition Letters* 30, 2 (2009), 88–97.
- [46] BUEHLER, M., IAGNEMMA, K., AND SINGH, S. **The DARPA urban challenge : autonomous vehicles in city traffic**, vol. 56. springer, 2009.

- [47] KRIZHEVSKY, A., AND HINTON, G. **Learning multiple layers of features from tiny images**. Tech. rep., Citeseer, 2009.
- [48] OVERETT, G., PETERSSON, L., ANDERSSON, L., AND PETTERSSON, N. **Boosting a heterogeneous pool of fast hog features for pedestrian and sign detection**. In *Intelligent Vehicles Symposium, 2009 IEEE* (2009), IEEE, pp. 584–590.
- [49] SNIDER, J. M. **Automatic steering methods for autonomous automobile path tracking**. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08* (2009).
- [50] TRADIŠAUSKAS, N., JUHL, J., LAHRMANN, H., AND JENSEN, C. S. **Map matching for intelligent speed adaptation**. *IET Intelligent Transport Systems* 3, 1 (2009), 57–66.
- [51] BASCÓN, S. M., RODRÍGUEZ, J. A., ARROYO, S. L., CABALLERO, A. F., AND LÓPEZ-FERRERAS, F. **An optimization on pictogram identification for the road-sign recognition task using svms**. *Computer Vision and Image Understanding* 114, 3 (2010), 373–383.
- [52] BELAROUSSI, R., FOUCHER, P., TAREL, J.-P., SOHEILIAN, B., CHARBONNIER, P., AND PAPARODITIS, N. **Road sign detection in images : A case study**. In *Pattern Recognition (ICPR), 2010 20th International Conference on* (2010), IEEE, pp. 484–488.
- [53] KANNAN, S., THANGAVELU, A., AND KALIVARADHAN, R. **An intelligent driver assistance system (i-das) for vehicle safety modelling using ontology approach**. *International Journal of Ubicomp* 1, 3 (2010), 15–29.
- [54] NAIR, V., AND HINTON, G. E. **Rectified linear units improve restricted boltzmann machines**. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (2010), pp. 807–814.
- [55] PAN, S. J., YANG, Q., AND OTHERS. **A survey on transfer learning**. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.
- [56] RUTA, A., LI, Y., AND LIU, X. **Robust class similarity measure for traffic sign recognition**. *IEEE Transactions on Intelligent Transportation Systems* 11, 4 (2010), 846–855.
- [57] ŠEGVIC, S., BRKIĆ, K., KALAFATIĆ, Z., STANISAVLJEVIĆ, V., ŠEVROVIĆ, M., BUDIMIR, D., AND DADIĆ, I. **A computer vision assisted geoinformation inventory for traffic infrastructure**. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on* (2010), IEEE, pp. 66–73.
- [58] YOUNG, K. L., REGAN, M. A., TRIGGS, T. J., JONTOF-HUTTER, K., AND NEWS-TEAD, S. **Intelligent speed adaptation—effects and acceptance by young inexperienced drivers**. *Accident Analysis & Prevention* 42, 3 (2010), 935–943.
- [59] GONZÁLEZ, Á., GARRIDO, M. Á., LLORCA, D. F., GAVILÁN, M., FERNÁNDEZ, J. P., ALCANTARILLA, P. F., PARRA, I., HERRANZ, F., BERGASA, L. M., SOTELO, M. Á. G., AND OTHERS. **Automatic traffic signs and panels inspection system using computer vision**. *IEEE Transactions on intelligent transportation systems* 12, 2 (2011), 485–499.
- [60] KRÄHENBÜHL, P., AND KOLTUN, V. **Efficient inference in fully connected crfs with gaussian edge potentials**. In *Advances in neural information processing systems* (2011), pp. 109–117.

- [61] LARSSON, F., AND FELSBURG, M. **Using Fourier descriptors and spatial models for traffic sign recognition**. In *Scandinavian Conference on Image Analysis* (2011), Springer, pp. 238–249.
- [62] LUNDQUIST, C., AND SCHÖN, T. B. **Joint ego-motion and road geometry estimation**. *Information Fusion* 12, 4 (2011), 253–263.
- [63] SERMANET, P., AND LECUN, Y. **Traffic sign recognition with multi-scale convolutional networks**. In *Neural Networks (IJCNN), The 2011 International Joint Conference on* (2011), IEEE, pp. 2809–2813.
- [64] TIMOFTE, R., AND VAN GOOL, L. **Sparse representation based projections**. In *Proceedings of the 22nd British machine vision conference-BMVC 2011* (2011), Citeseer, pp. 61–1.
- [65] ZEILER, M. D., TAYLOR, G. W., AND FERGUS, R. **Adaptive deconvolutional networks for mid and high level feature learning**. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE, pp. 2018–2025.
- [66] **Vehicle safety, deliverable 4.8u of the ec fp7 project dacota**. Available online : <http://www.dacota-project.eu/>, 2012. (accessed on 14 March 2017).
- [67] ALONSO, I. P., LLORCA, D. F., GAVILÁN, M., PARDO, S. Á., GARCÍA-GARRIDO, M. Á., VLACIC, L., AND SOTELO, M. Á. **Accurate global localization using visual odometry and digital maps on urban environments**. *IEEE Transactions on Intelligent Transportation Systems* 13, 4 (2012), 1535–1545.
- [68] CIREŞAN, D., MEIER, U., AND SCHMIDHUBER, J. **Multi-column deep neural networks for image classification**. *arXiv preprint arXiv :1202.2745* (2012).
- [69] GARCÍA-GARRIDO, M. A., OCANA, M., LLORCA, D. F., ARROYO, E., POZUELO, J., AND GAVILÁN, M. **Complete vision-based traffic sign recognition supported by an i2v communication system**. *Sensors* 12, 2 (2012), 1148–1169.
- [70] GASSER, T. M., AND WESTHOFF, D. **Bast-study : Definitions of automation and legal issues in germany**. In *Proceedings of the 2012 road vehicle automation workshop* (2012).
- [71] HANS, Z., SOULEYRETTE, R., AND BOGENREIF, C. **Horizontal curve identification and evaluation**. Tech. rep., 2012.
- [72] HUTH, V., BIRAL, F., MARTÍN, Ó., AND LOT, R. **Comparison of two warning concepts of an intelligent curve warning system for motorcyclists in a simulator study**. *Accident Analysis & Prevention* 44, 1 (2012), 118–125.
- [73] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. **Imagenet classification with deep convolutional neural networks**. In *Advances in neural information processing systems* (2012), pp. 1097–1105.
- [74] LAI, F., CARSTEN, O., AND TATE, F. **How much benefit does intelligent speed adaptation deliver : An analysis of its potential contribution to safety and environment**. *Accident Analysis & Prevention* 48 (2012), 63–72.
- [75] LI, Z., CHITTURI, M., BILL, A., AND NOYCE, D. **Automated identification and extraction of horizontal curve information from geographic information system roadway maps**. *Transportation Research Record : Journal of the Transportation Research Board*, 2291 (2012), 80–92.
- [76] MAURYA, A. K., AND BOKARE, P. S. **Study of deceleration behaviour of different vehicle types**. *International Journal for Traffic and Transport Engineering* 2, 3 (2012), 253–270.

- [77] MØGELMOSE, A., TRIVEDI, M. M., AND MOESLUND, T. B. **Vision-based traffic sign detection and analysis for intelligent driver assistance systems : Perspectives and survey.** *IEEE Trans. Intelligent Transportation Systems* 13, 4 (2012), 1484–1497.
- [78] PAPANODITIS, N., PAPELARD, J.-P., CANNELLE, B., DEVAUX, A., SOHEILIAN, B., DAVID, N., AND HOUZAY, E. **Stereopolis ii : A multi-purpose and multi-sensor 3d mobile mapping system for street visualisation and 3d metrology.** *Revue française de photogrammétrie et de télédétection* 200, 1 (2012), 69–79.
- [79] STALLKAMP, J., SCHLIPSING, M., SALMEN, J., AND IGEL, C. **Man vs. computer : Benchmarking machine learning algorithms for traffic sign recognition.** *Neural networks* 32 (2012), 323–332.
- [80] SZEPE, T., AND ASSAL, S. F. **Pure pursuit trajectory tracking approach : Comparison and experimental validation.** *International Journal of Robotics and Automation* 27, 4 (2012), 355.
- [81] TIELEMAN, T., AND HINTON, G. **Lecture 6.5-rmsprop : Divide the gradient by a running average of its recent magnitude.** *COURSERA : Neural networks for machine learning* 4, 2 (2012), 26–31.
- [82] GEIGER, A., LENZ, P., STILLER, C., AND URTASUN, R. **Vision meets robotics : The kitti dataset.** *The International Journal of Robotics Research* 32, 11 (2013), 1231–1237.
- [83] HOUBEN, S., STALLKAMP, J., SALMEN, J., SCHLIPSING, M., AND IGEL, C. **Detection of traffic signs in real-world images : The german traffic sign detection benchmark.** In *Neural Networks (IJCNN), The 2013 International Joint Conference on* (2013), IEEE, pp. 1–8.
- [84] LEE, J.-W., AND PRABHUSWAMY, S. **A unified framework of adaptive cruise control for speed limit follower and curve speed control function.** Tech. rep., SAE Technical Paper, 2013.
- [85] LIANG, M., YUAN, M., HU, X., LI, J., AND LIU, H. **Traffic sign detection by roi extraction and histogram features-based recognition.** In *Neural Networks (IJCNN), The 2013 International Joint Conference on* (2013), IEEE, pp. 1–8.
- [86] LIU, R., AND DUAN, J. **A path tracking algorithm of intelligent vehicle by preview strategy.** In *Control Conference (CCC), 2013 32nd Chinese* (2013), IEEE, pp. 5630–5635.
- [87] MATHIAS, M., TIMOFTE, R., BENENSON, R., AND VAN GOOL, L. **Traffic sign recognition—how far are we from the solution ?** In *Neural Networks (IJCNN), The 2013 International Joint Conference on* (2013), IEEE, pp. 1–8.
- [88] SALTI, S., PETRELLI, A., TOMBARI, F., FIORAIO, N., AND DI STEFANO, L. **A traffic sign detection pipeline based on interest region extraction.** In *Neural Networks (IJCNN), The 2013 International Joint Conference on* (2013), IEEE, pp. 1–7.
- [89] SCHARWÄCHTER, T., ENZWEILER, M., FRANKE, U., AND ROTH, S. **Efficient multi-cue scene segmentation.** In *German Conference on Pattern Recognition* (2013), Springer, pp. 435–445.
- [90] SERMANET, P., EIGEN, D., ZHANG, X., MATHIEU, M., FERGUS, R., AND LECUN, Y. **Overfeat : Integrated recognition, localization and detection using convolutional networks.** *arXiv preprint arXiv :1312.6229* (2013).

- [91] UIJLINGS, J. R., VAN DE SANDE, K. E., GEVERS, T., AND SMEULDERS, A. W. **Selective search for object recognition**. *International journal of computer vision* 104, 2 (2013), 154–171.
- [92] WANG, G., REN, G., WU, Z., ZHAO, Y., AND JIANG, L. **A robust, coarse-to-fine traffic sign detection method**. In *Neural Networks (IJCNN), The 2013 International Joint Conference on* (2013), IEEE, pp. 1–5.
- [93] WU, Y., LIU, Y., LI, J., LIU, H., AND HU, X. **Traffic sign detection based on convolutional neural networks**. In *Neural Networks (IJCNN), The 2013 International Joint Conference on* (2013), Citeseer, pp. 1–7.
- [94] YAMAMOTO, J., KARUNGARU, S., AND TERADA, K. **Japanese road signs recognition using neural networks**. In *SICE Annual Conference (SICE), 2013 Proceedings of* (2013), IEEE, pp. 1144–1150.
- [95] ZHANG, D., XIAO, Q., WANG, J., AND LI, K. **Driver curve speed model and its application to acc speed control in curved roads**. *International journal of automotive technology* 14, 2 (2013), 241–247.
- [96] CHEN, L.-C., PAPANDREOU, G., KOKKINOS, I., MURPHY, K., AND YUILLE, A. L. **Semantic image segmentation with deep convolutional nets and fully connected crfs**. *arXiv preprint arXiv :1412.7062* (2014).
- [97] COMMITTEE, S. O.-R. A. V. S., AND OTHERS. **Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems**. *SAE Standard J 3016* (2014), 1–16.
- [98] GEIGER, A., LAUER, M., WOJEK, C., STILLER, C., AND URTASUN, R. **3d traffic scene understanding from movable platforms**. *IEEE transactions on pattern analysis and machine intelligence* 36, 5 (2014), 1012–1025.
- [99] GIRSHICK, R., DONAHUE, J., DARRELL, T., AND MALIK, J. **Rich feature hierarchies for accurate object detection and semantic segmentation**. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 580–587.
- [100] GRUYER, D., BELAROUSSI, R., AND REVILLOUD, M. **Map-aided localization with lateral perception**. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE* (2014), IEEE, pp. 674–680.
- [101] HE, K., ZHANG, X., REN, S., AND SUN, J. **Spatial pyramid pooling in deep convolutional networks for visual recognition**. In *European conference on computer vision* (2014), Springer, pp. 346–361.
- [102] JIN, J., FU, K., AND ZHANG, C. **Traffic sign recognition with hinge loss trained convolutional neural networks**. *IEEE Transactions on Intelligent Transportation Systems* 15, 5 (2014), 1991–2000.
- [103] JO, K.-H., AND OTHERS. **Information retrieval of led text on electronic road sign for driver-assistance system using spatial-based feature and nearest cluster neighbor classifier**. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on* (2014), IEEE, pp. 531–536.
- [104] KINGMA, D. P., AND BA, J. **Adam : A method for stochastic optimization**. *arXiv preprint arXiv :1412.6980* (2014).
- [105] LEHTONEN, E., LAPPI, O., KOIRIKIVI, I., AND SUMMALA, H. **Effect of driving experience on anticipatory look-ahead fixations in real curve driving**. *Accident Analysis & Prevention* 70 (2014), 195–208.

- [106] LIN, T.-Y., MAIRE, M., BELONGIE, S., HAYS, J., PERONA, P., RAMANAN, D., DOLLÁR, P., AND ZITNICK, C. L. **Microsoft coco : Common objects in context**. In *European conference on computer vision* (2014), Springer, pp. 740–755.
- [107] LIU, C., CHANG, F., AND CHEN, Z. **Rapid multiclass traffic sign detection in high-resolution images**. *IEEE Transactions on Intelligent Transportation Systems* 15, 6 (2014), 2394–2403.
- [108] SHARIF RAZAVIAN, A., AZIZPOUR, H., SULLIVAN, J., AND CARLSSON, S. **CNN features off-the-shelf : an astounding baseline for recognition**. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (2014), pp. 806–813.
- [109] SIMONYAN, K., AND ZISSERMAN, A. **Very deep convolutional networks for large-scale image recognition**. *arXiv preprint arXiv :1409.1556* (2014).
- [110] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. **Dropout : a simple way to prevent neural networks from overfitting**. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
- [111] TIMOFTE, R., ZIMMERMANN, K., AND VAN GOOL, L. **Multi-view traffic sign detection, recognition, and 3d localisation**. *Machine vision and applications* 25, 3 (2014), 633–647.
- [112] YANG, Y., AND WU, F. **Real-time traffic sign detection via color probability model and integral channel features**. In *Chinese Conference on Pattern Recognition* (2014), Springer, pp. 545–554.
- [113] ZITNICK, C. L., AND DOLLÁR, P. **Edge boxes : Locating object proposals from edges**. In *European conference on computer vision* (2014), Springer, pp. 391–405.
- [114] BADRINARAYANAN, V., KENDALL, A., AND CIPOLLA, R. **Segnet : A deep convolutional encoder-decoder architecture for image segmentation**. *arXiv preprint arXiv :1511.00561* (2015).
- [115] BELL, S., UPCHURCH, P., SNAVELY, N., AND BALA, K. **Material recognition in the wild with the materials in context database**. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 3479–3487.
- [116] EVERINGHAM, M., ESLAMI, S. A., VAN GOOL, L., WILLIAMS, C. K., WINN, J., AND ZISSERMAN, A. **The pascal visual object classes challenge : A retrospective**. *International journal of computer vision* 111, 1 (2015), 98–136.
- [117] GIRSHICK, R. **Fast r-cnn**. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1440–1448.
- [118] HE, K., ZHANG, X., REN, S., AND SUN, J. **Delving deep into rectifiers : Surpassing human-level performance on ImageNet classification**. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1026–1034.
- [119] HOIEM, D., HAYS, J., XIAO, J., AND KHOSLA, A. **Guest editorial : Scene understanding**. *International Journal of Computer Vision* 112, 2 (2015), 131–132.
- [120] IOFFE, S., AND SZEGEDY, C. **Batch Normalization : Accelerating deep network training by reducing internal covariate shift**. *arXiv preprint arXiv :1502.03167* (2015).
- [121] LAU, M. M., LIM, K. H., AND OTHERS. **Malaysia traffic sign recognition with convolutional neural network**. In *Digital Signal Processing (DSP), 2015 IEEE International Conference on* (2015), IEEE, pp. 1006–1010.

- [122] LIU, W., RABINOVICH, A., AND BERG, A. C. **Parsenet : Looking wider to see better**. *arXiv preprint arXiv :1506.04579* (2015).
- [123] LONG, J., SHELHAMER, E., AND DARRELL, T. **Fully convolutional networks for semantic segmentation**. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 3431–3440.
- [124] PARK, M., LEE, S., AND HAN, W. **Development of steering control system for autonomous vehicle using geometry-based path tracking algorithm**. *ETRI Journal* 37, 3 (2015), 617–625.
- [125] QIAO, Y., CAPPELLE, C., AND RUICHEK, Y. **Place recognition based visual localization using lbp feature and svm**. In *Mexican International Conference on Artificial Intelligence* (2015), Springer, pp. 393–404.
- [126] REN, S., HE, K., GIRSHICK, R., AND SUN, J. **Faster r-cnn : Towards real-time object detection with region proposal networks**. In *Advances in neural information processing systems* (2015), pp. 91–99.
- [127] RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATY, A., KHOSLA, A., BERNSTEIN, M., AND OTHERS. **Imagenet large scale visual recognition challenge**. *International Journal of Computer Vision* 115, 3 (2015), 211–252.
- [128] SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCHE, V., AND RABINOVICH, A. **Going deeper with convolutions**. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1–9.
- [129] WANG, D., YUE, S., XU, J., HOU, X., AND LIU, C.-L. **A saliency-based cascade method for fast traffic sign detection**. In *Intelligent Vehicles Symposium (IV), 2015 IEEE* (2015), IEEE, pp. 180–185.
- [130] YU, F., AND KOLTUN, V. **Multi-scale context aggregation by dilated convolutions**. *arXiv preprint arXiv :1511.07122* (2015).
- [131] ZHENG, S., JAYASUMANA, S., ROMERA-PAREDES, B., VINEET, V., SU, Z., DU, D., HUANG, C., AND TORR, P. H. **Conditional random fields as recurrent neural networks**. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1529–1537.
- [132] BIAN, X., LIM, S. N., AND ZHOU, N. **Multiscale fully convolutional network with application to industrial inspection**. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on* (2016), IEEE, pp. 1–8.
- [133] BOJARSKI, M., DEL TESTA, D., DWORAKOWSKI, D., FIRNER, B., FLEPP, B., GOYAL, P., JACKEL, L. D., MONFORT, M., MULLER, U., ZHANG, J., AND OTHERS. **End to end learning for self-driving cars**. *arXiv preprint arXiv :1604.07316* (2016).
- [134] CHANGZHEN, X., CONG, W., WEIXIN, M., AND YANMEI, S. **A traffic sign detection algorithm based on deep convolutional neural network**. In *Signal and Image Processing (ICSIP), IEEE International Conference on* (2016), IEEE, pp. 676–679.
- [135] CORDTS, M., OMRAN, M., RAMOS, S., REHFELD, T., ENZWEILER, M., BENENSON, R., FRANKE, U., ROTH, S., AND SCHIELE, B. **The cityscapes dataset for semantic urban scene understanding**. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 3213–3223.



- [136] DAI, J., HE, K., AND SUN, J. **Instance-aware semantic segmentation via multi-task network cascades**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 3150–3158.
- [137] GÁMEZ SERNA, C., LOMBARD, A., RUICHEK, Y., AND ABBAS-TURKI, A. **Gps-based curve estimation for an adaptive pure pursuit algorithm**. In *Mexican International Conference on Artificial Intelligence* (2016), Springer.
- [138] HE, K., ZHANG, X., REN, S., AND SUN, J. **Deep residual learning for image recognition**. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.
- [139] JUNG, S., LEE, U., JUNG, J., AND SHIM, D. H. **Real-time traffic sign recognition system with deep convolutional neural network**. In *The 13th International Conference on Ubiquitous Robots and Ambient Intelligence* (2016), URAI.
- [140] LI, Y., MØGELMOSE, A., AND TRIVEDI, M. M. **Pushing the “speed limit” : high-accuracy us traffic sign recognition with convolutional neural networks**. *IEEE Transactions on Intelligent Vehicles* 1, 2 (2016), 167–176.
- [141] LIN, G., SHEN, C., VAN DEN HENGEL, A., AND REID, I. **Efficient piecewise training of deep structured models for semantic segmentation**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 3194–3203.
- [142] LIU, C., CHANG, F., AND LIU, C. **Occlusion-robust traffic sign detection via cascaded colour cubic feature**. *IET Intelligent Transport Systems* 10, 5 (2016), 354–360.
- [143] LIU, W., ANGUELOV, D., ERHAN, D., SZEGEDY, C., REED, S., FU, C.-Y., AND BERG, A. C. **Ssd : Single shot multibox detector**. In *European conference on computer vision* (2016), Springer, pp. 21–37.
- [144] LOMBARD, A., HAO, X., ABBAS-TURKI, A., EL MOUDNI, A., GALLAND, S., AND OTHERS. **Lateral control of an unmaned car using gnss positioning in the context of connected vehicles**. *Procedia Computer Science* 98 (2016), 148–155.
- [145] PASZKE, A., CHAURASIA, A., KIM, S., AND CULURCIELLO, E. **Enet : A deep neural network architecture for real-time semantic segmentation**. *arXiv preprint arXiv :1606.02147* (2016).
- [146] PINHEIRO, P. O., LIN, T.-Y., COLLOBERT, R., AND DOLLÁR, P. **Learning to refine object segments**. In *European Conference on Computer Vision* (2016), Springer, pp. 75–91.
- [147] QIAN, R., LIU, Q., YUE, Y., COENEN, F., AND ZHANG, B. **Road surface traffic sign detection with hybrid region proposal and fast r-cnn**. In *Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016 12th International Conference on* (2016), IEEE, pp. 555–559.
- [148] ROY, A., AND TODOROVIC, S. **A multi-scale cnn for affordance segmentation in rgb images**. In *European Conference on Computer Vision* (2016), Springer, pp. 186–201.
- [149] SPICHKOVA, M., SIMIC, M., SCHMIDT, H., CHENG, J., DONG, X., GUI, Y., LIANG, Y., LING, P., AND YIN, Z. **Formal models for intelligent speed validation and adaptation**. *Procedia Computer Science* 96 (2016), 1609–1618.

- [150] SZEGEDY, C., VANHOUCKE, V., IOFFE, S., SHLENS, J., AND WOJNA, Z. **Rethinking the inception architecture for computer vision**. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2818–2826.
- [151] TIAN, B., CHEN, R., YAO, Y., AND LI, N. **Robust traffic sign detection in complex road environments**. In *Vehicular Electronics and Safety (ICVES), 2016 IEEE International Conference on* (2016), IEEE, pp. 1–5.
- [152] WANG, S., DENG, Z., AND YIN, G. **An accurate gps-imu/dr data fusion method for driverless car based on a set of predictive models and grid constraints**. *Sensors* 16, 3 (2016), 280.
- [153] YANG, Y., LUO, H., XU, H., AND WU, F. **Towards real-time traffic sign detection and classification**. *IEEE Transactions on Intelligent Transportation Systems* 17, 7 (2016), 2022–2031.
- [154] YOUSSEF, A., ALBANI, D., NARDI, D., AND BLOISI, D. D. **Fast traffic sign recognition using color segmentation and deep convolutional networks**. In *International Conference on Advanced Concepts for Intelligent Vision Systems* (2016), Springer, pp. 205–216.
- [155] ZHU, Z., LIANG, D., ZHANG, S., HUANG, X., LI, B., AND HU, S. **Traffic-sign detection and classification in the wild**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 2110–2118.
- [156] ABDULLA, W. **Mask r-cnn for object detection and instance segmentation on keras and tensorflow**. [https://github.com/matterport/Mask\\_RCNN](https://github.com/matterport/Mask_RCNN), 2017.
- [157] ABEDIN, Z., DHAR, P., HOSSENAND, M. K., AND DEB, K. **Traffic sign detection and recognition using fuzzy segmentation approach and artificial neural network classifier respectively**. In *Electrical, Computer and Communication Engineering (ECCE), International Conference on* (2017), IEEE, pp. 518–523.
- [158] AGHDAM, H. H., HERAVI, E. J., AND PUIG, D. **A practical and highly optimized convolutional neural network for classifying traffic signs in real-time**. *International Journal of Computer Vision* 122, 2 (2017), 246–269.
- [159] CHILAMKURTHY, S. **Keras tutorial - traffic sign recognition**, 2017.
- [160] CHOLLET, F. **Xception : Deep learning with depthwise separable convolutions**. *arXiv preprint* (2017), 1610–02357.
- [161] GARCIA-GARCIA, A., ORTS-ESCOLANO, S., OPREA, S., VILLENA-MARTINEZ, V., AND GARCIA-RODRIGUEZ, J. **A review on deep learning techniques applied to semantic segmentation**. *arXiv preprint arXiv :1704.06857* (2017).
- [162] HE, K., GKIOXARI, G., DOLLÁR, P., AND GIRSHICK, R. **Mask r-cnn**. In *Computer Vision (ICCV), 2017 IEEE International Conference on* (2017), IEEE, pp. 2980–2988.
- [163] ISLAM, K. T., AND RAJ, R. G. **Real-time (vision-based) road sign recognition using an artificial neural network**. *Sensors* 17, 4 (2017), 853.
- [164] KYRIAKIDIS, M., DE WINTER, J. C., STANTON, N., BELLET, T., VAN AREM, B., BROOKHUIS, K., MARTENS, M. H., BENGLER, K., ANDERSSON, J., MERAT, N., AND OTHERS. **A human factors perspective on automated driving**. *Theoretical Issues in Ergonomics Science* (2017), 1–27.
- [165] LI, J., LIANG, X., WEI, Y., XU, T., FENG, J., AND YAN, S. **Perceptual generative adversarial networks for small object detection**. In *IEEE CVPR* (2017).

- [166] NEUHOLD, G., OLLMANN, T., BULÒ, S. R., AND KONTSCIEDER, P. **The mapillary vistas dataset for semantic understanding of street scenes**. In *ICCV* (2017), pp. 5000–5009.
- [167] SAADNA, Y., AND BEHLOUL, A. **An overview of traffic sign detection and classification methods**. *International Journal of Multimedia Information Retrieval* 6, 3 (2017), 193–210.
- [168] YUAN, Y., XIONG, Z., AND WANG, Q. **An incremental framework for video-based traffic sign detection, tracking, and recognition**. *IEEE Transactions on Intelligent Transportation Systems* 18, 7 (2017), 1918–1929.
- [169] ZHANG, J., HUANG, M., JIN, X., AND LI, X. **A real-time chinese traffic sign detection algorithm based on modified yolov2**. *Algorithms* 10, 4 (2017), 127.
- [170] ZHAO, H., SHI, J., QI, X., WANG, X., AND JIA, J. **Pyramid scene parsing network**. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 2881–2890.
- [171] ZHU, H., YUEN, K.-V., MIHAYLOVA, L., AND LEUNG, H. **Overview of environment perception for intelligent vehicles**. *IEEE Transactions on Intelligent Transportation Systems* 18, 10 (2017), 2584–2601.
- [172] ARCOS-GARCÍA, Á., ÁLVAREZ-GARCÍA, J. A., AND SORIA-MORILLO, L. M. **Deep neural network for traffic sign recognition systems : An analysis of spatial transformers and stochastic optimisation methods**. *Neural Networks* 99 (2018), 158–165.
- [173] ARCOS-GARCÍA, Á., ÁLVAREZ-GARCÍA, J. A., AND SORIA-MORILLO, L. M. **Evaluation of deep neural networks for traffic sign detection systems**. *Neurocomputing* 316 (2018), 332–344.
- [174] ARMINGOL, J. M., ALFONSO, J., ALIANE, N., CLAVIJO, M., CAMPOS-CORDOBÉS, S., DE LA ESCALERA, A., DEL SER, J., FERNÁNDEZ, J., GARCÍA, F., JIMÉNEZ, F., AND OTHERS. **Environmental perception for intelligent vehicles**. In *Intelligent Vehicles* (2018), Elsevier, pp. 23–101.
- [175] CHEN, L.-C., PAPANDREOU, G., KOKKINOS, I., MURPHY, K., AND YUILLE, A. L. **DeepLab : Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs**. *IEEE transactions on pattern analysis and machine intelligence* 40, 4 (2018), 834–848.
- [176] DELOITTE. **Connected and autonomous vehicles in ontario - implications for the insurance industry**. <https://www2.deloitte.com/content/dam/Deloitte/ca/Documents/consulting/ca-EN-CVAV-Research-Insurance-Report-2018Apr25-Final-AODA.PDF>, 2018.
- [177] EUROPEAN COMMISSION, DIRECTORATE-GENERAL FOR INTERNAL MARKET, I. E., AND SMES. **On the road to automated mobility : An eu strategy for mobility of the future**. <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52018DC0283&from=EN>, 2018.
- [178] GHIASI, G., LIN, T.-Y., AND LE, Q. V. **Dropblock : A regularization method for convolutional networks**. In *Advances in Neural Information Processing Systems* (2018), pp. 10748–10758.
- [179] HUANG, W., HUANG, M., AND ZHANG, Y. **Detection of traffic signs based on combination of gan and faster-rcnn**. In *Journal of Physics : Conference Series* (2018), vol. 1069, IOP Publishing, p. 012159.

- [180] LEE, H. S., AND KIM, K. **Simultaneous traffic sign detection and boundary estimation using convolutional neural network**. *IEEE Transactions on Intelligent Transportation Systems* (2018).
- [181] LI, J., AND WANG, Z. **Real-time traffic sign recognition based on efficient cnns in the wild**. *IEEE Transactions on Intelligent Transportation Systems*, 99 (2018), 1–10.
- [182] LUO, H., YANG, Y., TONG, B., WU, F., AND FAN, B. **Traffic sign recognition using a multi-task convolutional neural network**. *IEEE Transactions on Intelligent Transportation Systems* 19, 4 (2018), 1100–1111.
- [183] MELETIS, P., AND DUBBELMAN, G. **Training of convolutional networks on multiple heterogeneous datasets for street scene semantic segmentation**. In *2018 IEEE Intelligent Vehicles Symposium (IV)* (2018).
- [184] SERNA, C. G., AND RUICHEK, Y. **Classification of traffic signs : The european dataset**. *IEEE Access* (2018).
- [185] VAN BRUMMELEN, J., O'BRIEN, M., GRUYER, D., AND NAJJARAN, H. **Autonomous vehicle perception : The technology of today and tomorrow**. *Transportation research part C : emerging technologies* (2018).
- [186] YU, C., WANG, J., PENG, C., GAO, C., YU, G., AND SANG, N. **Bisenet : Bilateral segmentation network for real-time semantic segmentation**. In *European Conference on Computer Vision* (2018), Springer, pp. 334–349.
- [187] WALI, S. B., HANNAN, M. A., HUSSAIN, A., AND SAMAD, S. A. **Comparative survey on traffic sign detection and recognition : a review**.



# TABLE DES FIGURES

1.1	VaMoRs vehicle and a view of its interior. <sup>1</sup> . . . . .	4
1.2	Autonomous vehicles : adoptions and applications extracted from the report study [176]. . . . .	5
1.3	Overview of the autonomous navigation process. Light blue blocks indicate the five main components for autonomy. The light blue blocks with additional dotted boxes make reference to the components this thesis focus on for contributions. . . . .	7
2.1	Intraclass variability examples of European traffic signs. . . . .	16
2.2	European Traffic Sign Categories Definition. From left to right : main category, subcategories and most common shapes. . . . .	17
2.3	Examples of French directional signs. . . . .	18
2.4	Random representation of the 164 classes in the European traffic sign dataset. . . . .	20
2.5	Equipped UTBM car. . . . .	21
2.6	Screenshot of the Matlab Training Image Labeler tool used for the manual annotation. . . . .	22
2.7	Examples of some classes that contain intra class variability. Letters in the images are the initial of the country name they belong to. For example, B stands for Belgium, C for Croatia, etc. . . . .	25
2.8	Number of classes per category with less than 10 signs. . . . .	25
2.9	Number of images grouped by category and sub-category of the proposed European dataset. . . . .	26
2.10	Examples of challenging signs . . . . .	27
2.11	List of the 27 classes and their corresponding colors used for labeling. . . . .	30
2.12	Number of finely annotated pixels (y-axis) per class and their associated categories (x-axis). . . . .	31
2.13	Illustration of number of labeled instances per category and corresponding class. . . . .	32
2.14	Example of a captured image of UTBM-2 dataset and its corresponding semantic and instance labels. For the semantic classes refer to Fig. 2.11. Instance labels are numbered from 1 to n for each class but for visualization purposes they are presented with random colors. . . . .	33

2.15	Examples of traffic signs not included in the GTSRB. (a) shows four directional signs. (b) shows two examples containing additional panels. In the first example, the truck below the 30 speed limit sign, indicates that the restriction only applies for that type of vehicles ; while in the second example of (b), the first panel announces that the speed limit is only applicable for 600 m and the second panel indicates the timing applicable. The missing signs (c) make reference to curves, obstacles, barriers and limited access on the side. . . . .	35
2.16	Example of a GTSDb image and its corresponding semantic and instance labels. The semantic color class correspond to the Mapillary [166] definition. Instance labels are numbered from 1 to n, but for visualization purposes they are presented with random colors. . . . .	36
3.1	LeNet-5 Network architecture proposed by Lecun et al. [9] for digits recognition. . . . .	43
3.2	Network architecture proposed by Ciseran et al. [68]. (a) DNN architecture. (b) MCDNN architecture. (c) Training a DNN. The dataset is preprocessed before training (P block), then, at the beginning of every epoch, the images are distorted (D block). . . . .	43
3.3	Network architecture proposed by Aghdam et al. [158]. Light blue represents the convolution layers, green the ReLu activation layers, yellow the pooling layers, dark blue the fully connected layers and red the dropout layer	44
3.4	CNN architecture drawn from [181]. Light blue blocks represent the Convolution + BatchNormalization + ReLu layers, yellow the pooling layers, red the dropout layer, green in this case refers to layer concatenation, dark blue shows a fully-connected layer + BatchNormalization + ReLu while purple refers to dense layer with Softmax activation. The numbers indicated for the Convolution layers refer to the kernel sizes used. . . . .	45
3.5	Architecture drawn from Chilamkurthy proposal [159]. . . . .	46
3.6	Dropblock processing pipeline. The blue regions in the image representing the activation units, contain semantic information. Dropping continuous regions removes closely related information and consequently enforce remaining units to learn features every epoch for classifying the input image.	48
3.7	An example of some augmented traffic signs from the European dataset. . . . .	52
3.8	Visual representation of Class_CNN_Symmetric, Class_CNN and Class_CNN_RB. . . . .	54
3.9	Sign symbol classes grouped in Others category for the GTSRB and the ETSD. . . . .	56
3.10	Image size analysis for the incorrect predictions on the ETSD test set. Results are obtained with the CNN models trained with data-augmentation. . . . .	57
3.11	Random sample of incorrect predictions of the ETSD with the CNN_asymmetricK model trained with data-augmentation. . . . .	58
3.12	Random sample of incorrect predictions of the ETSD with the Improved_CNN_8-layers model trained with data-augmentation. . . . .	58

3.13	Random sample of incorrect predictions of the ETSD with our proposed Class_CNN model trained with data-augmentation. . . . .	59
4.1	Visual representation of the differences between object detection (objects enclosed with RoIs), semantic segmentation (dense pixel inference) and instance segmentation (dense pixel inference of the detected objects). . . .	64
4.2	System proposal pipeline composed of two modules : Semantic segmentation and Traffic sign recognition. . . . .	71
4.3	SegNet architecture extracted from [114]. . . . .	72
4.4	Overview of the PSPNet architecture [170]. The feature map in (b) corresponds to the map obtained after removing the fully connected layers of ResNet [138]. . . . .	73
4.5	DeepLab architecture [175]. . . . .	74
4.6	Overview of the Bilateral Segmentation Network (BiSeNet) extracted from [186]. . . . .	75
4.7	Pipeline of our proposed traffic sign recognition module. T.S. refers to Traffic Sign. The Mask R-CNN block represents the detection module while the refinement is composed of the RoI extraction and class score (S) filtering. The CNN classifier block makes reference to the classification module that will output a traffic sign class only if the predicted probability (P) is at least 0.9. . . . .	76
4.8	Mask R-CNN data flow. . . . .	77
4.9	CNN structure represented by blocks for the category and class classifiers. Block 1 and 2 (B1 & B2) are the same for both classifiers. . . . .	80
4.10	Urban/rural subset selection of the ETSD. (f) and (g) correspond to text-based signs while the rest make reference to symbol-based signs. . . . .	83
4.11	Semantic segmentation results on test samples of UTBM-2 dataset. . . . .	91
4.12	Precision-Recall curves of the detection module (Mask R-CNN) using 2 CNNs for category identification. Cat_CNN refers to the categories classifier defined in Table 4.1, while Class_CNN refers to the classes classifier defined in Table 3.1. . . . .	95
4.13	Comparison between AUC values of the common categories between the GTSDb and its extended version. . . . .	97
4.14	Example of detection and classification results by our proposed traffic sign recognition module on the GTSDb. Column (a) shows some images evaluated on the original GTSDb using Mask R-CNN + Class_CNN trained on the GTSRB to detect 43 classes. Column (b) refers to the detection with Mask R-CNN on the extended version of the GTSDb + Urban_Class_CNN trained in the urban set of the ETSD to detect 132 classes. . . . .	102



4.15	Example of detection and classification results by our proposed traffic sign recognition module on UTBM-2 dataset. The recognition is performed with Mask R-CNN + Urban_Class_CNN trained in the urban set of the ETSD to detect 132 classes. The left column shows image examples with recognition accuracy above 75% while the right column shows images with traffic sign recognition accuracy under 75%. . . . .	103
4.16	Incorrect predictions after detection performed on the GTSDb with the Class_CNN trained on the GTSRB. . . . .	104
4.17	Incorrect predictions after detection performed on the GTSDb extended version with the Urban_Class_CNN trained on the selected urban set of the ETSD. . . . .	104
4.18	Incorrect predictions after detection performed on the UTBM-2 dataset with the Urban_Class_CNN trained on the selected urban set of the ETSD. . . . .	104
5.1	Overall work-flow of the vehicle path tracking system. Blue dotted bounding box highlights our contributions. . . . .	107
5.2	Illustration of Pure Pursuit geometry [49]. . . . .	110
5.3	Illustration of Stanley steering controller [28]. . . . .	111
5.4	Illustration of Alice steering controller [40]. . . . .	112
5.5	Illustration of Lombard steering controller [144]. . . . .	113
5.6	Overall data-flow of the Dynamic Speed Adaptation module. . . . .	114
5.7	Speed limit driving scenario . . . . .	116
5.8	Curve parameters representation . . . . .	117
5.9	Curve classification . . . . .	118
5.10	Curve classification of UTBM-2 dataset. Sharp curves are marked with red circles. . . . .	119
5.11	Centripetal force diagram . . . . .	120
5.12	Curve driving scenario. . . . .	121
5.13	Curve driving scenario projecting vehicle position into the reference path. . . . .	122
5.14	Datasets with sharp curves and speed limits marked. Normal black line represents a speed limit of 50 km/h, green line a speed limit of 30 km/h and sharp curves are in red. . . . .	124
5.15	Root mean square error ( $e_{rms}$ ) comparison of lateral errors obtained in 30 km/h speed limit segments. . . . .	126
5.16	Average root mean square error ( $\overline{e_{rms}}$ ) of lateral error comparison between methods for 30 km/h speed limit segments. . . . .	127
5.17	Root mean square error ( $e_{rms}$ ) comparison of lateral errors obtained in sharp curves for each dataset. . . . .	128
5.18	Speed and lateral error comparison in UTBM-2 dataset using PP method with and without DSA. . . . .	129

5.19 Average root mean square error ( $\overline{e_{rms}}$ ) of lateral error comparison between methods for sharp curves. . . . . 129



# LISTE DES TABLES

1.1	BASt, NHTSA and SAE levels of driving automation. System refers to the driver assistance system, combination to driver assistance systems, or automated driving systems, as appropriate. . . . .	6
2.1	Relation of classes and total number of traffic signs by country. . . . .	26
2.2	Number of semantic and instance classes annotated in the UTBM-2 dataset and its corresponding conversion to different formats (CamVid, CityScapes, Mapillary). 'Original' makes reference to the number of classes that the original dataset format has, while 'Added' means the number of classes not included in the dataset but considered for annotation as extra ones. . .	32
2.3	Number of signs by category for the training and testing sets of each dataset. The Others category in the original GTSDb and its extended version do not represent the same classes (refer to [83] for the classes in the original GTSDb and see Fig. ?? for the extended GTSDb) but we put it like that for the name convention. The unknown category contains traffic signs labeled which class does not correspond to any of the considered ones. . .	37
3.1	Architecture of our CNN for traffic sign class identification (Class.CNN). 'chan' makes reference to the number of channels, 'kS' to kernel size, 'std' to stride and 'kpProb' to the probability of keeping the original feature. . . .	47
3.2	Accuracy percentage results obtained on the GTSRB and European test sets. The input size refers to image "width×height×channels", while the number of parameters is presented in millions (M) and time in milliseconds (ms). . . . .	49
3.3	Accuracy percentage results obtained on the GTSRB test set. The input size refers to image "width×height×channels", while the number of parameters is presented in millions (M) and time in milliseconds (ms). . . . .	53
3.4	Class_CNN accuracy percentage results obtained on the GTSRB test set. .	54
3.5	Accuracy percentage results obtained on the GTSRB and European test sets without and with performing data-augmentation. . . . .	55
3.6	Error percentage predictions by category on the GTSRB and ETSD test sets. Results are computed from the CNN models trained with data-augmentation. . . . .	56
4.1	Architecture of our CNN for traffic sign categories identification (Cat.CNN). 'chan' makes reference to the number of channels, 'kS' to kernel size, 'std' to stride and 'kpProb' to the probability of keeping the original feature. . . .	80

4.2	Quantitative comparison of deep networks for semantic segmentation on the UTBM-2 dataset. The training time corresponds to 100 epochs while the testing time to the average per image. . . . .	89
4.3	mIoU quantitative comparisons of semantic segmentation models on the UTBM-2 test set for 15 classes. . . . .	90
4.4	Classifiers' accuracy results and characteristics trained on the proposed Class_CNN and Cat_CNN described in Section 4.3.2.2. . . . .	92
4.5	Detection results on the GTSDB test set for 0.5 IoU (AUC values) . . . . .	95
4.6	Detection results on the extended GTSDB and UTBM-2 test sets using MaskRCNN+Urban_Class_CNN for 0.5 IoU (AUC values) . . . . .	96
4.7	Classification evaluation for German and French roads with the GTSDB, the extended GTSDB and UTBM-2 test sets. The evaluation is performed on the detected signs extracted by Mask R-CNN + the Urban Class_CNN. . . . .	98
5.1	Error comparison in zones of 30 km/h with different Lateral Control methods. Results are expressed in meters. . . . .	126
5.2	Error comparison in sharp curves of different Lateral Control methods for the Simulated path. Results are expressed in meters. . . . .	127
5.3	Error comparison in sharp curves of different Lateral Control methods for UTBM-2 dataset. Results are expressed in meters. . . . .	128
5.4	Error comparison in sharp curves of different Lateral Control methods for UTBM-3 dataset. Results are expressed in meters. . . . .	132

# IV

## ANNEXES



# A

## PUBLICATIONS

### A.1/ CONFERENCES

1. C. Gámez Serna, A.Lombard, Y. Ruichek, and A. Abbas-Turki. **Gps-Based Curve Estimation for an Adaptive Pure Pursuit Algorithm**. In *Mexican International Conference on Artificial Intelligence* pp. 497-511. Springer, Cham, 2016.

### A.2/ JOURNALS

1. C. Gámez Serna, and Y. Ruichek. **Dynamic Speed Adaptation for Path Tracking Based on Curvature Information and Speed Limits**. *Sensors* 17, no. 6 (2017) : 1383. [pdf]
2. C. Gámez Serna, and Y. Ruichek. **Classification of Traffic Signs : The European Dataset**. *IEEE Access* (2018). [pdf][project]
3. C. Gámez Serna, and Y. Ruichek. **Traffic Signs Detection and Classification for European Urban Environments**. *IEEE Transactions on Intelligent Transportation Systems* (2019). Under review.





