Universidad de Alcalá

PhD. Program in Information and Communications Technologies

Prediction of Vehicle Intentions for Advanced Autonomous Driving

Author

Rubén Izquierdo Gonzalo

Advisors

Dr. D. Miguel Ángel Sotelo Vázquez

Dr. D. David Fernández Llorca

Alcalá de Henares, 21st of September, 2020

Tempus fugit

Agradecimientos

Es de bien nacido ser agradecido, y ha llegado el momento de agradecer. Esto significa que al fin he terminado la tesis. Parecía que este momento no iba a llegar nunca. Heme aquí escribiendo las que serán las primeras líneas de mi tesis, aunque en realidad son las últimas que estoy escribiendo. Me esmeraré, necesito dar una buena impresión a los lectores. La mayoría solo leerá estas líneas y ojeará las figuras por encima, pero la primera impresión es la que cuenta.

Mi más sincero agradecimiento a mis tutores los doctores D. Miguel Ángel Sotelo Vázquez y D. David Fernández Llorca. Gracias por todo el apoyo que me habéis brindado durante los ocho años que llevo en éste nuestro laboratorio. Aún recuerdo el día que recibí un correo del profesor de la asignatura de control ofreciéndome una beca de colaboración. Y pensé... de entre todas las personas de la universidad me han elegido a mi! Ahí comenzó mi viaje hacia lo que aún no sabía sería una tesis doctoral. Durante estos ocho años ha habido momentos buenos y mejores, pero también los ha habido regulares. Al final todo ha terminado en buen puerto y eso es gracias a vosotros.

En el apartado *tutores en la sombra* Nacho merece una mención especial. Siempre involucrado en el día a día y en el "bajo nivel". Nunca le ha negado la ayuda a un pobre industrial con las cosas de los telecos. Gracias Nacho.

Como miembro de mayor antigüedad[1] del grupo INVETT, a excepción de los jefes, me veo en la obligación de hacer un especial reconocimiento a la labor de coaching que todos los miembros del grupo realizáis a diario. En especial al núcleo duro. No diré nombres pero todos sabéis que son, por orden de llegada: Kñao, MH, MA y AQP. Aprovecho estas líneas que se que vais a leer para desearos fuerza para lo que os queda por delante. [2]

[1] Aquellos que abandonaron la escuela o se cambiaron de grupo no conservan la antigüedad.
[2] Nótese lo mal que lo he pasado escribiendo este párrafo. Temía ser el desgüe de las tesis.

En el apartado de agradecimientos dulces no podía desaprovechar esta ocasión para agradecerle a Noe que nos endulce las tardes, habitualmente de los lunes, con sus postres. Tampoco puedo dejar pasar la oportunidad de pedir más postres. Aquí los jefes deberían tomar nota, a más azúcar, más le damos al coco.[3]

Gracias a Cristina, mi novia. Por aguantarme todos estos años, trece si empezamos a contar desde el principio y ocho desde que empezaron los quebraderos de cabeza de la investigación. Gracias en especial por aguantarme este último año de redacción de tesis, que además, se ha juntado con la cuarentena y la denominada nueva normalidad. Por si todo esto no fura suficiente, también ha sido el primer año de convivencia. Pese a todo, creo que al final el esfuerzo ha merecido la pena para ambos. Lo que la tesis ha unido, que no lo separe el hombre.

Gracias a mis padres. Si los errores de un hijo son los fracasos de sus padres, los aciertos tiene que ser también sus éxitos. Enhorabuena por que habéis acertado de pleno. No haber contribuido de forma directa a la consecución de esta tesis no os quita el merito de la misma. Me habéis criado y me habéis dado una educación que es lo mínimo que hace falta para sacar matrícula de honor en primero de la vida. Cuando me preguntéis si voy a terminar de estudiar ya podré decir que si.

Gracias a mis amigos, tomar unas cañas después de un día duro de trabajo siempre viene bien para alegrar el cuerpo y el espíritu. El tiempo pasa, y por si no lo recordabais este erá el último año.

No podía terminar los agradecimientos sin agradecerle a esta tesis todo lo que me ha aportado. Aprender he aprendido un montón, pero eso se le presupone a una tesis doctoral. Esta tesis me ha permitido participar en congresos nacionales e internacionales con los que he recorrido literalmente medio mundo y parte del otro, desde Hawaii hasta Nueva Zelanda. También he tenido la oportunidad de participar en proyectos y competiciones. Una vez casi ganamos uno de coches autónomos ¿te imaginas? Darle las gracias a la tesis es una forma de reconocerme a mi mismo todo el esfuerzo que he realizado para poder llegar donde me encuentro hoy. A toro pasado volvería sin dudarlo a aceptar aquella beca de colaboración que en su día me ofrecieron y que inició todo esto.

[3]Una vida saludable requiere de una dieta variada y ejercicio físico

Resumen

Durante los últimos años el interés por los sistema de predicción avanzada de trayectorias de vehículos y de intenciones ha crecido notablemente. Inicialmente la predicción tanto de trayectorias como de maniobras se ha centrado en observaciones realizadas desde puntos estáticos tales como la infraestructura. Esto se ha debido a la falta de bases de datos adecuadas para la predicción desde un punto de vista centrado en el vehículo.

Esta tesis aborda el problema de la predicción de maniobras y trayectorias en entornos de autopistas con un enfoque basado en aprendizaje máquina. Ante la ausencia de bases de datos apropiadas para su desarrollo se tomó la decisión de realizar una base de datos específica para la predicción tanto de trayectorias como de maniobras. Así nace *The PREVENTION dataset*. Una base de datos grabada desde la perspectiva de un vehículo que incluye cerca de 6 horas de grabaciones. Cuenta con 2 cámaras, un láser rotativo y 3 radares, además de un sistema de localización diferencial y una unidad de medida inercial. Esta base de datos incluye numerosas anotaciones manuales que permiten identificar vehículos y cambios de carril además de las posiciones de los vehículos.

El sistema de predicción de maniobras se basa en un arquitectura de redes neuronales convolucionales que clasifica una imagen de entrada en tres posibles categorías correspondientes con las acciones de cambio de carril a la izquierda y a la derecha y la acción de continuar en el carril actual. La imagen de entrada consta de tres canales en los que cada uno cumple una función. El canal rojo representar el entorno, la apariencia de la escena. El canal azul se emplea para seleccionar el objetivo de la predicción, del cual se dibuja el contorno actual y los pasados con diferentes niveles de intensidad creando una especie de estela que muestra la dinámica del vehículo. El canal verde se emplea para dibujar las estelas del resto de vehículos que actúan como elementos condicionantes de la acción realizada por el vehículo representado en el canal azul. Esta representación no limita el numero de vehículos en la escena y el numero de muestras que se pueden representar es de 255.

Para poder comparar el rendimiento del sistema de predicción de intenciones con la capacidad humana de predicción se ha realizado un estudio que evalúa la capacidad de predecir o detectar cambios de carril, así como la tasa de acierto de estos. Las métricas usadas para comparar el desempeño tanto de las personas como del sistema de predicción son la tasa de acierto y la anticipación. f El sistema de predicción de trayectorias adapta una red neuronal convolucional desarrollada para la clasificación de células en imágenes clínicas. Esta red extrae características a diferentes niveles de profundidad para finalmente generar una imagen de salida. La red ha sido modificada para tomar a la entrada una imagen 3D que codifica una secuencia de imágenes de un solo canal. La salida es similar a la entrada solo que codifica la misma secuencia en el futuro. Los vehículos son representados sobre una vista de pájaro que genera una representación gráfica de la escena. Además, los elementos como las líneas dibujadas en la carretera se pueden añadir en esta representación. La red es capaz de aprender la mecánica subyacente de las interacciones entre vehículos y el entorno para generar las posiciones de esos mismos vehículos en el futuro.

Para poder comparar los resultados obtenidos se ha implementado un sistema de predicción básico basado en un filtro de Kalman con un modelo de velocidad constante como línea de partida. La predicción de trayectorias se ha evaluado utilizando varias métricas comunes en la literatura, tales como el *RMSE*, *MAE*, *ATE* y el *FTE*.

Palabras clave: PREVENTION, Predicción de Maniobras, Predicción de Trayectorias, Aprendizaje Máquina, Factores Humanos.

Abstract

During the last years, the interest in advanced vehicle trajectory and intention prediction systems has grown remarkably. Initially, the prediction of both trajectories and maneuvers has been focused on observations made from static points of view, such as the infrastructure because of the lack of appropriate vehicle-centered datasets.

This thesis addresses the problem of predicting maneuvers and trajectories in highway environments with a machine learning approach. A specific database for the prediction of both trajectories and maneuvers was created because of the lack of appropriate ones. Thus, *The PREVENTION dataset* was born. A database recorded from an onboard perspective that includes almost 6 hours of recordings. It has 2 cameras, a rotating laser, and 3 radars, as well as a differential localization system and an inertial measurement unit. This database includes several manual annotations that allow the identification of vehicles and lane changes as well as the positions of the vehicles.

The maneuver prediction system is based on a convolutional neural network architecture that classifies an input image into three possible categories corresponding to left and right lane-change actions and the lane-keeping action. The input image consists of three channels, each one has a specific purpose. The red channel represents the environment, the appearance of the scene. The blue channel is used to select the prediction target, from which the current and past contours are drawn with different intensity levels creating a kind of trail that shows the dynamics of the vehicle. The green channel is used to draw the trails of all the surrounding vehicles that actuate as conditioning elements for the vehicle represented in the blue channel. This representation does not limit the number of vehicles in the scene and the number of samples that can be represented is 255.

To compare the performance of the intention prediction system with the human prediction capacity, a study was carried out to evaluate the capacity to predict or detect lane changes, as well as the lane change accuracy rate. The metrics used to compare the performance of people and the prediction system are the accuracy rate and the anticipation.

The trajectory prediction system adapts a convolutional neural network developed for the classification of cells in clinical images. This network extracts features at different depth levels to finally generate an output image. The network has been modified to take a 3D input image that encodes a single-channel image sequence. The output is similar to the input, but it encodes the same sequence in the future. The vehicles are represented on a bird's eye view that generates a graphic representation of the scene. Besides, elements such as road markings can be added to this representation. The network learns the underlying mechanics and interactions between vehicles and the environment to generate the positions of those vehicles in the future.

A Kalman filter with a constant speed model has been implemented as a baseline to compare with the obtained results. Trajectory prediction has been evaluated using several common metrics in the literature, such as *RMSE*, *MAE*, *ATE* and *FTE*.

KeyWords: PREVENTION, Maneuver Prediction, Trajectory Prediction, Machine Learning, Human Factors.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ACC	Adaptive Cruise Control.
ADAS	Advanced Driver Assistance System.
AEB	Automatic Emergency Braking.
AES	Autonomous Emergency Steering.
ANN	Artificial Neural Network.
ATE	Average Trajectory Error.
AUC	Area Under the Curve.
BEV	Bird Eye View.
BN	Bayesian Network.
BRAVE	BRidging gaps for the adoption of Automated VEhicles.
BTU	British Thermal Unit.
CAN	Controller Area Network.
CAS	Collision Avoidance System.
CBR	Case-Base Reasoning.
CNN	Convolutional Neural Network.
CVAE	Conditional Variational Autoencoder.
DBN	Dynamic Bayesian Network.
DGNSS	Differential Global Navigation Satellite System.
EKF	Extended Kalman Filter.
FOV	Field Of View.
FTE	Final Trajectory Error.

GAN	Generative Adversarial Network.
GDP	Gross Domestic Product.
GHG	Greenhouse gas.
GMM	Gaussian Mixture Models.
GPNN	Gaussian Process Neural Network.
GPS	Global Positioning System.
GPU	Graphics Processing Unit.
GVT	Ground Vehicle Teleoperated.
H3D	Honda 3D Dataset.
HDD	Honda Driving Dataset.
IMU	Inertial Measurement Unit.
INS	Inertial Navigation System.
IoU	Intersection over Union.
ITS	Intelligent Transportation System.
KF	Kalman Filter.
LCE	Lane Change Event.
LiDAR	Light Detection and Ranging.
LLC	Left Lane Change.
LSTM	Long Short-Term Memory.
MAE	Mean Absolute Error.
NLC	No Lane Change.
NTP	Network Time Protocol.
PPS	Pulse Per Second.
PREVENTION	PREdicion of VEhicle inteNTION.
RANSAC	RANdom SAmple Consensus.
RF	Radio Frequency.
RLC	Right Lane Change.

RMSE	Root Mean Squared Error.
RNN	Recurrent Neural Network.
ROI	Region of Interest.
RTK	Real Time Kinematic.
SVD	Singular Value Decomposition.
SVM	Support Vector Machine.
TTC	Time to Collision.
V2I	Vehicle to Infrastructure.
V2V	Vehicle to Vehicle.
VUT	Vehicle Under Test.

Chapter 1

Introduction

1.1 Context Analysis

The transport sector brings many benefits to society and the economy. It is a critical sector in the economic and social development of countries. The European Commission estimates the transport sector accounts for about 5% of *Gross Domestic Product (GDP)*. The transport sector accounts for 5,6% in the US, achieving a more than $1 trillion in GDP. However, the transport sector is also responsible for many fatalities as well as massive emissions of *Greenhouse gas (GHG)*.

Mobility demand is continuously growing; in 2018 it reached 4.7 trillion kilometers only in the US. This number is expected to grow up to 5,6 trillion kilometers in 2050. Freight transport demand is expected to grow by 52% from 397 billion in 2018 to 967 billion kilometers in 2050 as a result of economic development according to [1]. According to the World Health Organization [2], with current levels of mobility, approximately 1.35 million people died in 2018 as a result of road traffic crashes. Developed countries with high-incomes own 40% of the world's vehicles, but only 7% of the world's fatalities take place in these countries.

In contrast, low-income countries own 1% of the world's vehicles and 13% of total deaths caused by traffic accidents. This data reveals how access to technologies can help save lives. Road traffic crashes also cause economic losses equivalent to 3% of the gross domestic product on property damage. In the EU, the number of fatalities in highways represents only 8%, but an accident on a highway usually ends with disastrous consequences.

Worldwide GHG emissions by the transport sector represent 20% of the world's emissions. In the Euro area, this fraction rises to 29%

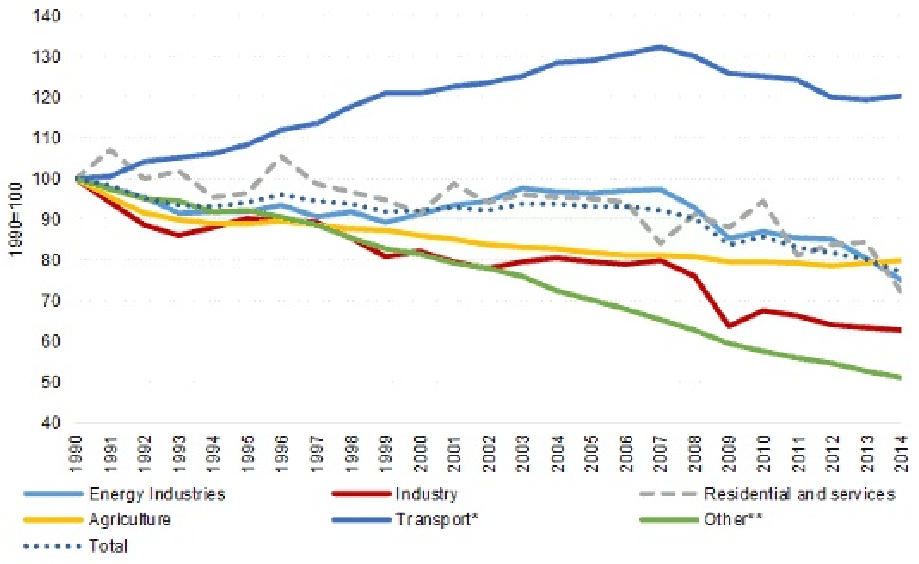


Figure 1.1: EU CO_2 emissions by sector 1990 - 2014. * Transport includes international aviation but excludes international maritime. ** Other include fugitive emissions from fuels, waste management and indirect CO_2 emissions.

according to [3]. The US Department of Energy's Office of Energy Efficiency and Renewable Energy estimates that light vehicles and medium and heavy trucks and buses consume more than 80% of the energy used by the transport sector. This consumption represents more than 20 quadrillions of *British Thermal Unit (BTU)* equivalent to 1,7 Gt of CO_2 only in the US. Transport emissions reached 1.1 Gt of CO_2 in the EU in 2017. Figure 1.1 shows the evolution of greenhouse emissions by sector in the EU zone relative to 1990 levels. All sectors have experienced a significant reduction except for the transport sector. Transport sector emissions were growing until the global financial collapse in 2008 when it fell to the 1999 level. The current level of economic development is pushing the transport sector emissions up.

On the other hand, the climate crisis has set the focus on GHG. The Paris agreement set an ambitious and global plan with the objective to reduce GHG emissions and achieve average global warming below 2 °C to the preindustrial level. The level of GHG must achieve a reduction between 80% and 95% in 2050 in the EU area [4]. Figure. 1.2 shows EU GHG emissions since 1990. The emissions have been decreasing continuously since 1990. 2020 objective with 20% reduction is currently accomplished but, 2030 target with 40% reduction seems challenging

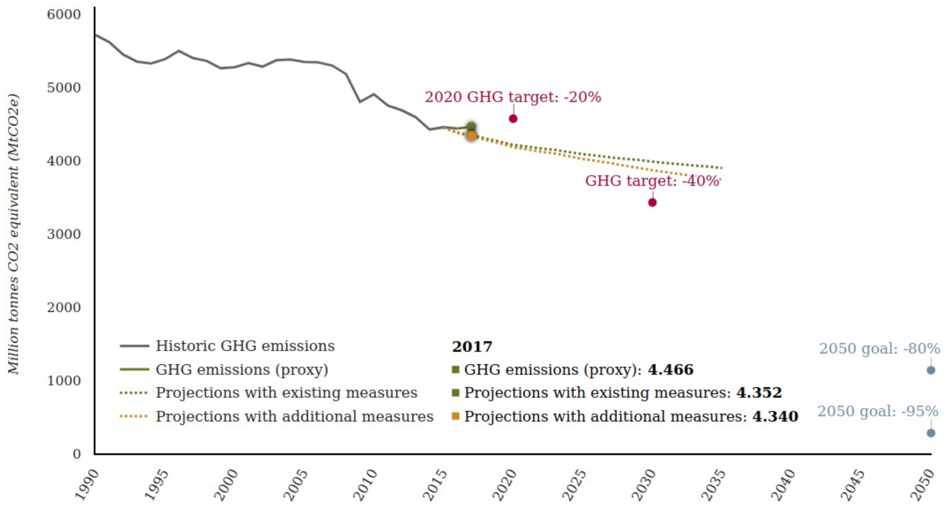
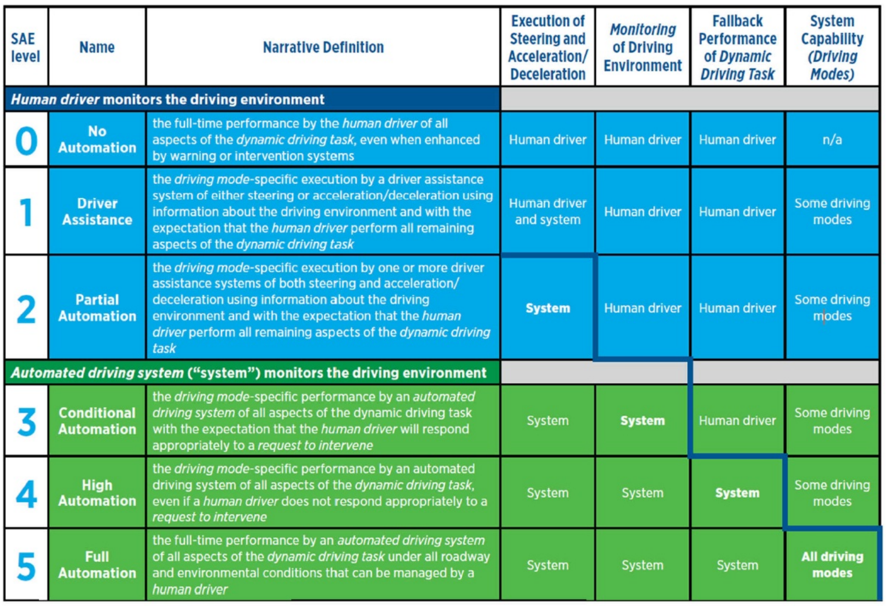


Figure 1.2: EU total CO_2 emissions 1990 - 2050.

with the current trend. An important effort reducing GHG emissions needs to be made by all the involved parts; governments, industries, services, and users.

The number of vehicles on the planet has reached 1.3 billion in 2016, according to [5]. These have doubled with respect to 2003 levels. Economic development, especially in developing countries, has brought vehicles to as many people as never before. This fact has produced a dramatic increase in traffic jams in almost all cities around the world. This problem cost \$305 billion in the United States alone, where every American lost 97 hours in traffic jams in 2018, according to [6]. The cost of road congestion in Europe is estimated to be €110 billion a year in 2012, according to [7].

The area of Intelligent Transportation Systems and Intelligent vehicles has a vital role to play building tomorrow's automotive paradigm facing the problems stated before. Autonomous vehicles are close to being a reality on the road in the next years. Autonomous vehicles have the capacity and the obligation to address the problems created by the transport sector. Autonomous vehicles can drive efficiently by removing any irrational motivation from the decision-making process, unlike humans. They can sense precisely the environment and act accordingly in a fraction of a second. They can drive continuously, needing no stops to rest, are not affected by distractions, and eliminating fatigue. Autonomous vehicles can communicate between themselves and the infrastructure using *Vehicle to Vehicle (V2V)* or *Vehicle to Infras-*



SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	No Automation	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
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 driver and system	Human driver	Human driver	Some driving modes
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 driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
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 to a *request to intervene*	System	**System**	Human driver	Some driving modes
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
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**

Figure 1.3: SAE - Automation levels definition.

tructure (V2I) communications, being able to know the state of the traffic at virtually everywhere, outperforming sensor ranges, and even human perception.

The capacity of autonomous vehicles is determined by a convention based on a reasoned agreement that logically describes the taxonomy of the autonomous vehicle at five different levels. This SAE J3016TM definition is not a specification and does not impose requirements, figure 1.3 shows a description of each automation level.

The reference case for automation is level 0, which is a non-automated manual driven vehicle.

- Level 1 - *Driver Assistance.* Enables driver assistance on lateral or longitudinal control using some environment information. The human driver and the assistance system works together to complete the driving action. Examples of level-1 automation are lane-keeping or adaptive cruise control systems.

- Level 2 - *Partial Automation.* Characterized by assistance in both longitudinal and lateral control. The human driver monitors the environment and supervises the driving task. Examples of level-2 automation are lane-keeping and adaptive cruise control.

- Level 3 - *Conditional Automation.* The vehicle is driven in an automated way performing lateral and longitudinal control. The human driver must be ready to take control as requested by the system. Level-3 automation example is traffic jam chauffeur.

- Level 4 - *High Automation.* The system performs all the driving functions under certain conditions. The possibility of removing driving actuators characterizes this level. The driver may have the option to control the vehicle if actuators do exist. Examples of level-4 are driverless taxis in restricted areas.
- Level 5 - *Full Automation.* The automated driving system is capable of performing all the driving tasks under all conditions. Human attention or intervention is not required.

Nowadays, commercial vehicles have reached automation level 3. Manufacturers have developed different systems to perform automated driving tasks. Tesla launched the autopilot system in October 2014, offering semi-autonomous driving and parking capabilities, the current version of autopilot has unexpectedly reduced its original capabilities. Toyota launched the Safety Sense system as part of its strategy to develop high-end safety systems in March 2015. Nissan's models equip the ProPilot system, which develops lane-keeping and *Adaptive Cruise Control (ACC)* with hands-on-wheel. Hands-off functions are only available in Japan since May of 2019. In February 2017, Volvo introduced hands-off ACC and lane-keeping functions in some of their high-end models under the label PilotAssist. VW group enabled the Traffic Jam Assist system in July 2017. This system performs ACC and lane-keeping under 55 kph with hands-off-wheel. Mercedes' autonomous assistance pack is denominated Driving Assistance Plus and performs hands-off lane-keeping and ACC at no speed limits.

Autonomous vehicles will share the road with human-driven vehicles for a long period of time. During this period autonomous vehicles will take advantage of V2V communications sharing their information and trajectories between them. However, human-driven cars with no predefined trajectories will be a source of uncertainty.

1.2 Motivation

Automated vehicles became a reality in the '80s with two research projects: the EUREKA project and the Autonomous Land driven Vehicle project. Both bring automated cars that can drive at limited speed, using a sort of sensors in restricted areas. At that point, interactions with non-automated vehicles did not represent a real challenge. Nowadays, commercial vehicles have reached automation level 3 and have become part of our lives. They will replace human-driven vehicles in a distant future, but both will coexist for a long period.

In the meanwhile, automated vehicles will share the road with manually driven cars. In this scenario, different behaviors and interaction will take place between automated and non-automated vehicles. Automated vehicles could share their trajectories between them and actuate in a coordinated manner. However, non-automated vehicles cannot share their trajectories or intentions because they are self-generated at the moment the driver reveals them.

In this scenario, automated vehicles need to deal with uncertainties relative to manually driven vehicles while planning their trajectories. Prediction becomes a pivotal ability to understand what other traffic agents will do even if they do not communicate their intentions. Humans also make predictions and incorporate them into their decision making. Human drivers are affected by some limitations such as distractions, reaction time, and tiredness, or fatigue. However, humans still are the best driving machines.

Graphics Processing Unit (GPU) computation has reached levels that enable complex image processing in real-time. The possibility to understand images, even video sequences faster and better than humans, brings the opportunity to predict the evolution of traffic scenes overcoming human reasoning. Automated vehicles can take advantage of their prediction capabilities by anticipating conditioning situations. Driving performance can be enhanced in terms of efficiency and safety, resulting in smoother traffic flow and fewer blocking situations. For this reason, a novel trajectory and maneuver prediction system based on image sequences and sensor data is presented. This new ability brings higher standards of safety and efficiency to automated vehicles.

1.3 Applications

The proposed models can be applied to predict trajectories and actions on highway scenarios. The scope of these models is not only autonomous vehicles but also human-driven vehicles.

Autonomous vehicles can improve their trajectories by using predictive motion or action models. Moreover, automated vehicles below automation level 4 must be in contact with the driver all the time. Providing the proper information to the driver is of utmost importance to reduce the driver's stress. Representation of future actions or trajectories of surrounding vehicles would help the driver to understand ego-vehicle behavior.

Non-automated vehicles can take advantage of these models using them as an input of their *Advanced Driver Assistance System (ADAS)*, such as collision warning, collision avoidance, or ACC.

In both cases, these models help to improve safety and driving efficiency through higher fuel efficiency and higher traffic flow. This contribution helps to reduce the three main problems generated by the transport sector: traffic congestion, GHG, and injuries or fatalities.

1.4 Document Outline

After the introduction in Chapter 1, Chapter 2 reviews in depth the most relevant works and datasets that address the trajectory or maneuver prediction problem.

Chapter 3 presents The PREVENTION Dataset, a dataset explicitly built to develop the work presented in this Ph.D. dissertation and fulfill identified shortages in this research area. Sensor setup, calibration, and synchronization mechanisms are detailed in this chapter, as well as the generated metadata information.

Chapter 4 presents a social study conducted to evaluate human capacity to anticipate lane changes in sequences extracted from The PREVENTION Dataset. This study sets a baseline based on human capabilities.

Chapter 5 describes the two learning-based predictive models developed. One focuses on lane change prediction based on vehicle motion and prediction target integration in an RGB image. The other one focuses on surrounding vehicles' trajectory prediction by integrating vehicle detection and road information as image sequences in a *Bird Eye View (BEV)* representation.

The results of the proposed algorithm are presented and discussed in Chapter 6. Finally, Chapter 7 contains the conclusions, main contributions and future research lines.

Chapter 2

State of the Art

In this chapter, the state of the art is reviewed in detail. This includes a review of the vehicle trajectory and maneuver prediction works as well as the available datasets employed to develop these works.

In the first section, the available datasets are reviewed analyzing three aspects: the acquisition point of view, sensor setup, and availability of data. An important distinction must be done regarding the location of the sensors, which can be on board a vehicle or from an extrinsic point of view. The first type has the disadvantage of being affected by occlusions. However, the results achieved when using these datasets do not change at deploy time. The second type is very valuable for understanding and evaluating the motion and behavior of vehicles and drivers under different traffic scenarios. However, they cannot be fully applied to onboard applications.

The second section reviews vehicle trajectory and vehicle intention prediction work with a special emphasis on the prediction target. The prediction target could be the ego vehicle or surrounding vehicles. Datasets that provide measure from an exterior point of view, such as infrastructure acquisition systems or drones have special consideration. The ego vehicle does not exist, consequently there is not a clear distinction between ego vehicle and surrounding vehicles' predictions. The main difference is that the data is not affected by occlusions as it is when recording with onboard sensors. They cannot be considered strictly ego vehicle centered because data are not recorded from this point of view, but according to the availability of surrounding vehicle information. This could be the most similar definition.

2.1 Datasets

This section reviews the available datasets on which most of the vehicle trajectory research or action-based predictions are based. Some of them are publicly available but others were developed for a specific purpose and the authors did not consider publishing them openly to the scientific community.

- NGSIM I80 dataset [8]
  The NGSIM program was launched by the U.S. Federal Highway Administration in 2005 to provide a knowledge base for developing micro and macroscopic driving models for traffic flow optimization.

  The researchers collected detailed data on the vehicle's trajectories along I-80 in the San Francisco Bay Area. The study area was approximately 500 meters long and consisted of a six-lane highway, including a high occupancy lane and an entrance lane. Seven synchronized cameras, mounted on top of an adjacent building, recorded vehicles passing through the area. The vehicle routes in the images are converted into coordinates in the road reference system, providing accurate positioning of the lane level and locations relative to other vehicles with a sampling rate of 10 Hz. This dataset has a total duration of 45 minutes, divided into three 15-minute parts. These periods represent the build-up of congestion and total congestion during the peak period.

  In addition to vehicle trajectory data, the I-80 dataset also contains computer-aided design and GIS files, aerial orthorectified photos, highway loop detector data in and around the study area, raw and processed video, signal synchronization settings on adjacent arterial roads, traffic signal information and locations, weather data, and aggregate data analysis reports. The full I-80 dataset is freely available at the NGSIM Web site at http://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm

- NGSIM HW101 dataset [9].
  The NGSIM HW101 is the second part of the NGSIM program. Following the same goal, on June 15, 2005, researchers collected detailed vehicle trajectory data on US 101, also known as the Hollywood Freeway, in Los Angeles, CA.

  The study area was approximately 640 meters long and consisted of a five-lane highway. An entry/exit lane was also present in

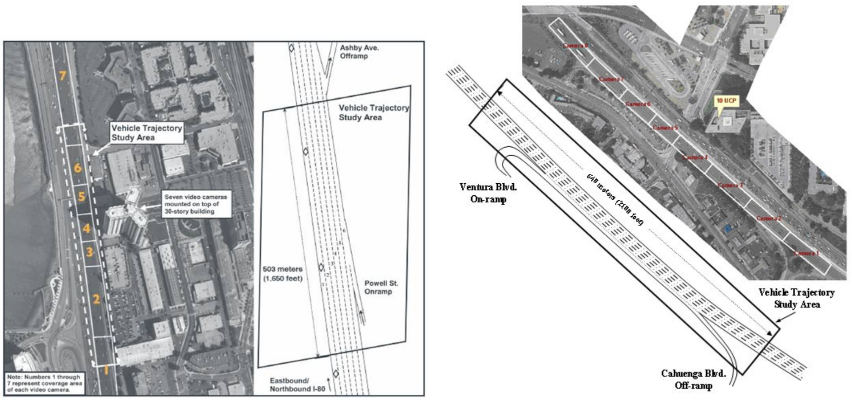

Vehicle Trajectory Study Area
Seven video cameras mounted on top of 30-story building
Note: Numbers 1 through 7 represent coverage area of each video camera.
Ashby Ave. Offramp
Vehicle Trajectory Study Area
503 meters (1,650 feet)
Powell St. Onramp
Eastbound/ Northbound I-80
Ventura Blvd. On-ramp
640 meters (2100 feet)
Vehicle Trajectory Study Area
Cahuenga Blvd. Off-ramp
10 UCP

Figure 2.1: NGSIM I80 and HW101 Recording Areas.

Figure 2.2: Tracking examples of KITTI Tracking Dataset.

the experiment area. Eight synchronized digital video cameras, mounted from the top of the 36-story building next to the highway (as shown in figure 2.1), recorded vehicles passing through the study area. The trajectories of the vehicles in the images are transformed into lane-level trajectories with precise positioning and relative location to the vehicles. The dataset is built with three 15-minute parts from 7:50 a.m. to 8:35 a.m. The available data is the same as the NGSIM I80 dataset.

- KITTI dataset [10], [11]
  The KITTI benchmark suite has provided different types of datasets for many research purposes. The object tracking benchmark can be used for trajectory prediction. There are eight different classes labeled, but only cars and pedestrians are used for this benchmark. A total of 50 sequences compose the whole dataset, which has 21 training sequences and 29 test sequences. For each sequence, 2D and 3D bounding boxes of pedestrians and vehicles are labeled in the image plane and in the point cloud. The goal of this dataset is to solve the tracking problem between frames, but the ground truth can be used to predict vehicles or pedestrians' trajectories.

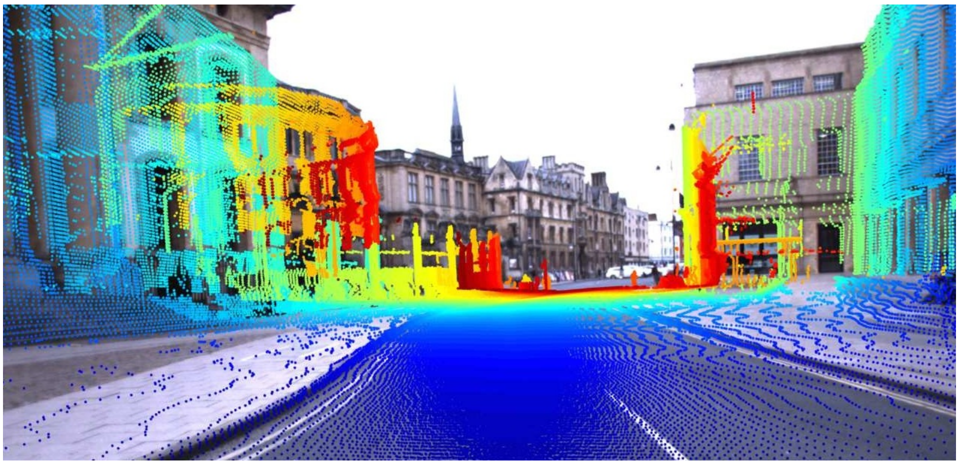


Figure 2.3: RobotCar data example.

- Oxford RobotCar dataset [12]
  The Oxford RobotCar dataset contains over 100 repetitions of a consistent route through Oxford city over a period of a year. The dataset captures many different combinations of weather, traffic status, vehicles, and pedestrians along the time with long-term changes such as construction and roadworks. The recording platform is a vehicle equipped with six cameras, *Light Detection and Ranging (LiDAR)*, GPS, and *Inertial Navigation System (INS)*. The primary purpose of this dataset is the development of long-term localization algorithms. This dataset provided a vast amount of data that could be used for trajectory prediction in urban scenarios. However, there is no information beyond the raw data.

- PKU dataset [13]
  The dataset published by the University of Peking contains data recorded on Beijing's fourth ring road for 97 minutes and over 69 km. The mobile platform has a GPS-IMU system for global location tasks and four HOKUYO LiDAR sensors (two long and two short-range) to measure the position of the surrounding vehicles and the road boundaries. The LiDAR range is 40 and 20 meters for the long and short-range, respectively. Positioning and environment measuring have different data rates, the ego vehicle location has a frequency close to 20 Hz, and the data coming from the LiDAR is approximately 10 Hz. The data set presents four types of records. The ego-motion log contains orientation (roll, pitch, and yaw), global positioning (north and east), and speed

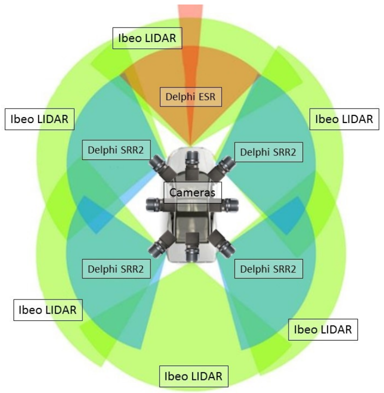


Figure 2.4: LISA-A testbed.

(north and east). Surrounding vehicle detections include position (x and y in local and global), speed (x and y in local and global), and dimensions (width and length). Road boundaries are recorded as 2D points (x and y) defining the limits of the road in local and global reference systems. Also, the ego lane changes are recorded, including the type (left or right), the start and end time of the maneuver, and the initial condition of the ego vehicle in the maneuver (x, y, and heading). There is no lane level information in this dataset.

- LISA-A dataset [14]
  The LISA-A dataset is a dataset with more than 100 hours of real-world recordings including a mobilEye sensor, eight cameras, six LiDARs, five radars, and GPS/acIMU sensors. The mobilEye sensor provides parameters of the lane structure. The raw data is available for all of the sensors. The metadata extracted from the raw measures consists of labels that identify each vehicle in each camera with a bounding box. No information about maneuvers or road positioning of vehicles is available. The actual downloadable file of the dataset includes only four sequences 100 seconds length approximately. The biggest part of the 100 hours dataset is not publicly available at present.

- ApolloScape dataset [15]
  The ApolloScape dataset was released in 2018, and it has been adding content since its release. The ApolloScape includes some datasets for different research topics. One of them is oriented to trajectory prediction problems.

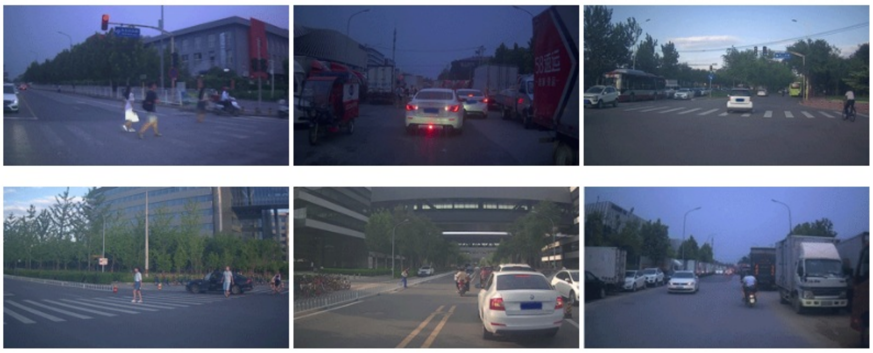


Figure 2.5: Scenes example of Apollo Trajectory Prediction dataset.

This dataset contains trajectories manually labeled using camera and LiDAR sensors. The data was collected in the city of Beijing in urban-scenarios mostly. They offer a sample of approximately 100 minutes of data with highly complicated traffic scenarios mixing vehicles, riders, and pedestrians. The frame rate of the provided annotations is at 2 Hz, which is insufficient to depict precisely the motion of vehicles in nonlinear trajectories such as lane changes or turn maneuvers.

The trajectory files are represented as 1-minute sequences, which include annotations for each frame with a unique id for each vehicle, object type, position, size, and orientation. This dataset is focused on trajectory prediction only, and action-oriented predictions such as lane changes cannot be applied due to the lack of road or lane structure information.

- Berkeley DeepDrive BDD100K [16]
  Berkeley DeepDrive is a video-based dataset with more than 100,000 sequences of videos recorded in many different hours, weather conditions, and traffic scenarios. This dataset was recorded for more than 50,000 drivers, making it useful to develop models with a variety of ego-vehicle behaviors. The images are 720p at 30 Hz complemented with *Global Positioning System (GPS)* location and *Inertial Measurement Unit (IMU)* data. This dataset provides high-level information such as lane markings, drivable area, and object detections on 100,000 images, one for each sequence. Additionally, instance segmentation is provided for a tenth of them. As far as only one image is labeled

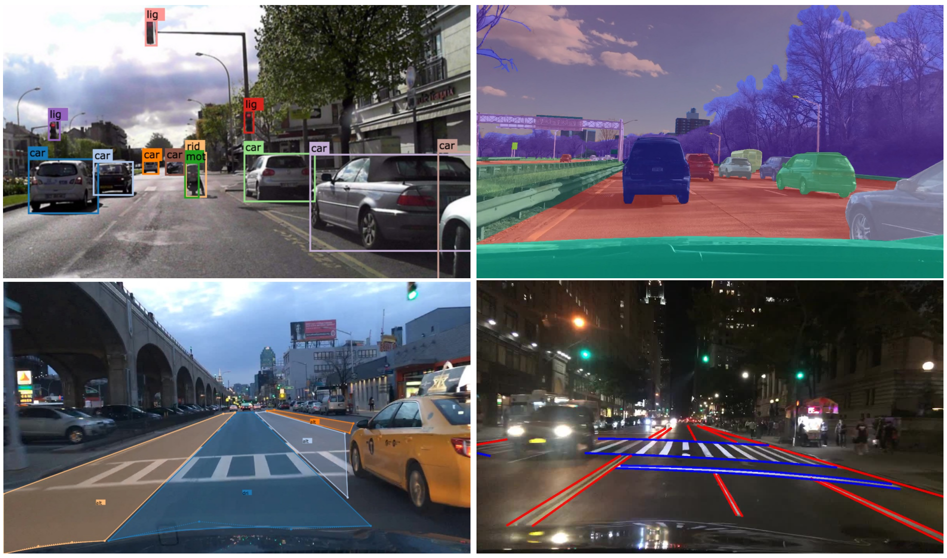


Figure 2.6: DeepDrive dataset features representation.

in each sequence, this dataset cannot be used for time-based algorithms until they provide fully labeled sequences. However, it has a huge potential to develop vision-based algorithms due to the vast amount of data.

- *Honda Driving Dataset (HDD)* [17]
  The HDD is a challenging dataset to enable research on learning driver behavior in real-life environments. Three cameras and one LiDAR are employed to sense the environment generating all-around 3D information and a visual representation of the middle front of the vehicle. A GPS with *Real Time Kinematic (RTK)* capability and an IMU complete the sensor setup. More than 100 hours of data were recorded in the San Francisco Bay area. The dataset is oriented to develop action-based predictions. Events are labeled in four different dimensions, goal-oriented action, stimulus-driven action, cause, and attention. However, the scope of these annotations is limited to the ego vehicle.

  The *Honda 3D Dataset (H3D)* [18] is an extension from the HDD with 3D tracking information, more than 1 million of 3D bounding boxes are labeled in 160 scenes representing more than 27,000 frames with pedestrians and interactive traffic. The labels are annotated at a rate of 2 Hz and linearly propagated to generate 10 Hz labels. This data rate is not enough to develop trajectory or

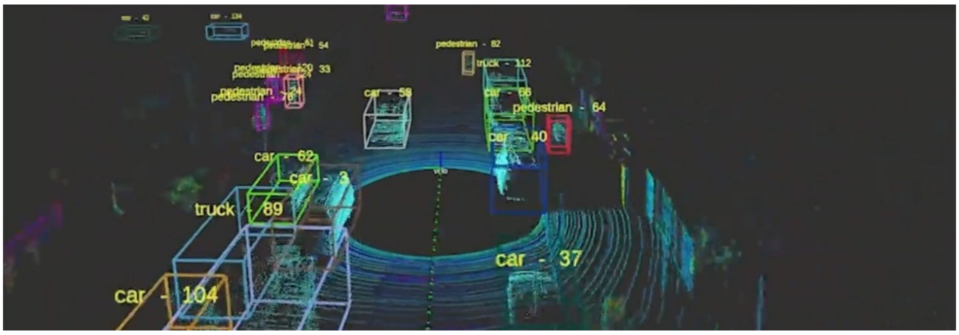


Figure 2.7: Tracking examples of HDD.

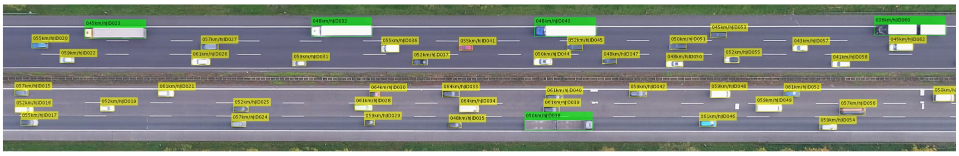


Figure 2.8: Recording Area HighD dataset.

maneuver prediction algorithms as it happens with ApolloScape dataset.

- HighD dataset [19]
  The highD data set is a new set of naturalistic vehicle trajectory data registered on German highways. Using a drone, the typical limitations of established traffic data collection methods, such as occlusions, are overcome by aerial perspective. Traffic was recorded in six different locations and includes more than 110500 vehicles. The trajectory of each vehicle, including its type, size, and maneuvers, is automatically extracted. Using state of the art computer vision algorithms, the positioning error is typically less than ten centimeters. Although the data set was created for the safety validation of highly automated vehicles, it is also suitable for many other tasks such as traffic pattern analysis or parameterization of driver models.

  The data available in the HighD dataset are positions of 110500 equivalent to 44500 kilometers traveled with a standard error below 10 centimeters and maneuvers conducted, such as lane changes. Positions, speed, and statistics such as time headway and *Time to Collision (TTC)* are also provided.

- Argoverse dataset [20]

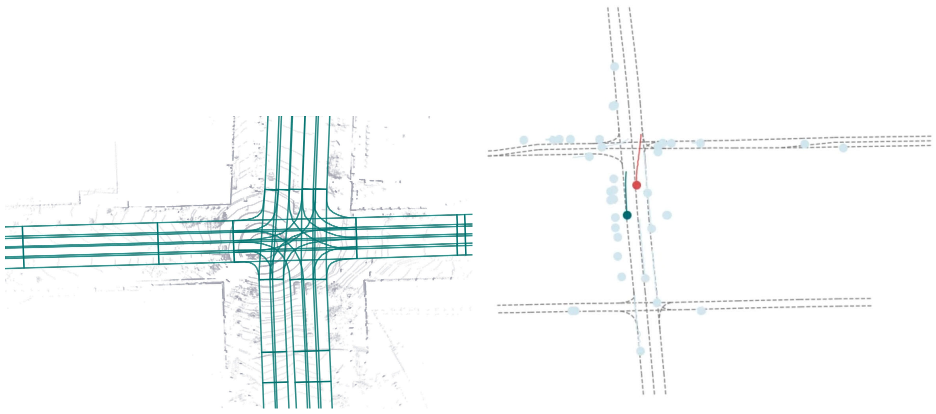


Figure 2.9: Example of Argoverse motion forecasting dataset.

The Argoverse dataset is a dataset recorded in the cities of Miami and Pittsburgh. High-definition digital maps of the involved area are available providing precise lane level information, drivable area, and ground height. The sequences were recorded with a vehicle equipped with 2 LiDARs, GPS, a 7 cameras rig with 360 degrees coverage running at 30 Hz, and a stereo camera system running at 5 Hz. Labels of 3D detections are provided for 113 sequences with more than 10,000 tracked objects. This data is oriented to develop tracking algorithms but can be also used to develop trajectory and maneuver predictions. The Argoverser motion forecasting is a collection of more than 320,000 sequences each 5 seconds long. Each sequence consists of a 2D bird-eye view position of each tracked object sampled at 10 Hz. The sequences represent lane changes, pass through intersection and vehicles taking left and right turns.

- Waymo dataset [21]
  The dataset contains independently-generated labels from LiDAR and camera data. LiDAR 3D bounding boxes are provided for vehicles, pedestrians, cyclists, and traffic sings. The detections are represented as 7-DOF bounding boxes in the vehicle's reference frame. Each detection has a unique ID identifier. The same objects are also labeled in the camera images providing 2D bounding boxes with a unique ID without correspondence with LiDAR IDs. This dataset provides 1000 segments, each one with 20 seconds of driving with 3D bounding boxes labels. Camera 2D bounding box labels are only provided for 100 of those segments.

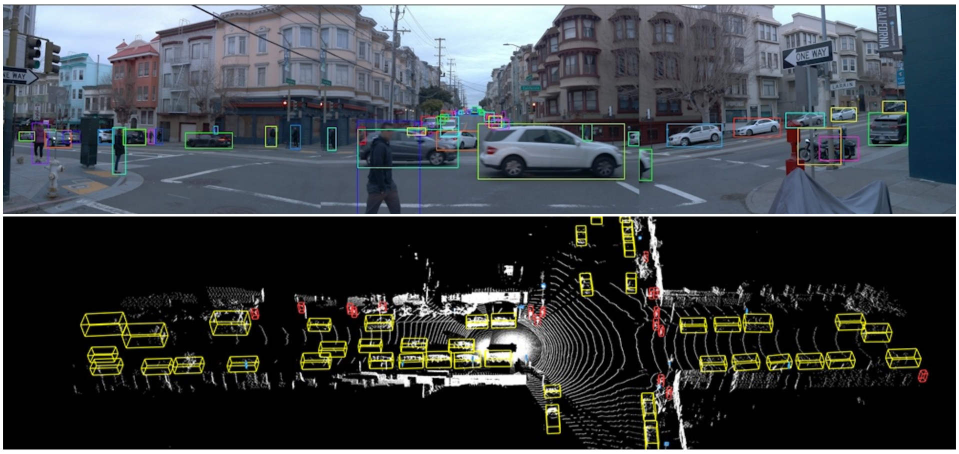


Figure 2.10: Waymo dataset example.

The datasets presented in this section are summarized in table 2.1. Regarding the perspective from which the data was recorded, two types can be distinguished basically: in-vehicle and third point of view such as infrastructure or drones. *Ego* and *Top* labels denote the in-vehicle and third point of view, respectively. The sensor setup is marked using a tick mark (✓) or an asterisk symbol ($*$). The tick mark represents that the sensor data is available. The asterisk symbol shows that the sensor data is not available but has been used to compute some information.

Static recording systems from a top view perspective have many advantages over in-vehicle recording systems. They are unaffected by occlusions and provide a complete understanding of the scene. Drivers do not know they are being recorded, and their behavior is not altered. A static recording system captures more vehicles than mobile platforms, but the vehicles travel a shorter distance and are visible for a shorter period. The most challenging problem when an algorithm is developed using this type of dataset is implementing it on in-vehicle applications. Datasets recorded from onboard sensor have the advantage of being directly deployed, and there is not a gap between development and deployment. On the contrary, the data are affected by occlusions, and there is a level of uncertainty that is not present in datasets recorded from heights.

If we take a look at the sensors used to build these datasets and consequently used in the autonomous vehicles that will develop autonomous tasks, we can find three types: camera, LiDAR, and radar.

The most common are cameras, followed by LiDARs, and lastly, radars. However, radars are the most common sensor in nowadays vehicles. NGSIM and Berkeley Deep Driver datasets are only based on image systems. Others, such as Honda 3D and PKU are based on LiDAR. Most of them use a combination of camera-based system and LiDAR solutions. However, only the LISA-A dataset provides radar detections, which are one of the best choices between expensive LiDARs or complex stereo camera systems for object detection, in spite of its robustness. Nowadays, brand new cars are commonly equipped with radar sensors as an essential element for proactive security systems such as *Automatic Emergency Braking (AEB)* or *Collision Avoidance System (CAS)*. Also, low-autonomy tasks such as speed planning are in charge of radars in the ACC systems. Datasets recorded from a mobile platform are usually equipped with a GPS system for global positioning tasks, and they are often complemented with an IMU. On the opposite side, the HighD and the NGSIM datasets are both recorded from an external and static point of view. Their measurements are referred to a static road reference system, and it does not need global positioning.

Datasets are built to fulfill a shortage in a specific field or research topic. For some fields or research topics, a massive amount of raw sensor data is enough, but the most valuable part of a dataset is the metadata or annotations generated by experts to label a specific circumstance. The metadata of each dataset has been carefully reviewed to establish the useful information they provide to develop predictive trajectory and action works. The Oxford RobotCar only provides raw sensor data because its primary purpose is the long-term vision and LiDAR-based localization. Berkeley Deep Drive is based only on a monocular camera system that is not able to generate range measurements; for this reason, trajectories cannot be generated. All the other datasets provide trajectories in a cartesian-metric reference system. Some such as KITTI, H3D, ApolloScape, and LISA-A provide trajectories in the ego-vehicle reference system, others such as NGSIM or HighD provide trajectories in the road reference system due to their specific point of view. The Argoverse dataset provides trajectories in a semi-global map-based reference system with an arbitrary origin of coordinates. Finally, the PKU dataset provides trajectories in both local and global reference system.

Lane level information is necessary for a precise scene understanding in urban and highway scenarios. Lanes establish dependencies between vehicles, especially in highway scenarios, by creating virtual walls that can be crossed or not, depending on the situation. Lane level information endows driving relationships between vehicles establishing roles such as lane following, overtaking, or cut-in and cut-out. The lane-level information allows for an easier understanding of vehicle trajectories by simplifying curved trajectories to straight ones in the lane reference system. KITTI, PKU, ApolloScape, H3D, and Waymo datasets do not provide lane-level information. NGSIM and HighD datasets neither provide lane-level information, but they were recorded in straight stretches of highways, and the trajectories are intrinsically referred to the lane reference system. Argoverse dataset does not provide lane detections but provides a detailed map of the recording area built with the lane's axes.

Only HDD and HighD datasets have event labels. HDD labeled the events made by the ego vehicle and the cause of this event, i.e., a stop-action caused by congestion. The events present in the HighD dataset are the number of lane changes developed for each vehicle during the time while crossing the recording area. More complex event annotations such as maneuver classification are promised to be available soon. Potentially, lane-change maneuvers can be easily computed if the lane information and the trajectories are available. The lane change event takes place when the center of the vehicle crosses the line between two lanes. The beginning of the lane change is much more relevant than the lane change event, and it is harder to define.

The most extended dataset is the Oxford RobotCar dataset with more than 200 hours of recordings. However, there are no detections nor annotations. The Argoverse motion forecasting dataset presents more than 300,000 sequences 5 seconds length of surrounding vehicle positions. However, 5 seconds of data are sometimes not long enough for motion recognition and long-term predictions. Following this line, Waymo dataset provides fewer samples but with a longer duration. H3D, Berkeley Deep Drive, and Apolloscape present trajectories labeled at an insufficient frame rate, 2 Hz in the best case. LISA-A and KITTI dataset has appropriate frame rates (10 Hz), but they have a minimal amount of data. Finally, the PKU dataset has an appropriate data rate and length but, lane-level information is not available. We can highlight HighD between the top-view datasets for its length compared with NGSIM datasets. These static datasets can be affected by an underlying problem. The area covered by them varies from 420 to 640 meters. This area could be driven at highway speed in between 12 and 25 seconds, depending on the length and the road congestion.

Dataset	Release date	Length	Rate	View	Image	LiDAR	Radar	GNSS-RTK	Events	Lanes	Trajectories
NGSIM I80	Dec 2006	3 seq @ 15 min	10 Hz	Top	✓					✓	✓
NGSIM HW101	Jan 2007	3 seq @ 15 min	10 Hz	Top	✓					✓	✓
KITTI	June 2013	50 seq	10 Hz	Ego	✓	✓		RTK			✓
Oxford RobotCar	Nov 2016	100 seq @ 2 h	33/10 Hz	Ego	✓	✓		GPS			
PKU	Jan 2017	97 min	10 Hz	Ego		$*$		GPS			✓
LISA-A	Sept 2017	4 seq @ 100 sec	30/10 Hz	Ego	✓	✓	✓	GPS		✓	✓
ApolloScape	Mar 2018	100 min	2 Hz	Ego	✓	✓		GPS			✓
Berkeley DeepDrive	May 2018	100K @ 1 frame	30 Hz*	Ego	✓			GPS		✓	
H3D	June 2018	160 seq	2 Hz	Ego		✓		GPS	†		✓
HighD	Oct 2018	16 h	25 Hz	Top	$*$				†	✓	✓
Argoverse Motion Forecasting	June 2019	320K seq @ 5 sec	10 Hz	Ego		$*$		GPS		✓	✓
Waymo	Aug 2019	1000 seq @ 20 sec	10 Hz	Ego	✓	✓		RTK			✓
PREVENTION$^\Delta$	Oct 2019	11 seq @ 30 min	10 Hz	Ego	✓	✓	✓	RTK	✓	✓	✓

Table 2.1: Datasets Overview

($*$) represents the sensor data is not available but has been used to compute some information.
(†) represents features that are not provided but can be computed from existing data.
(Δ) PREVENTION dataset is part of the contributions of this thesis.

2.2 Trajectory and Lane Change Prediction

In this section, state of the art related to trajectory and maneuver prediction problems is reviewed in detail. There are several essential points to take into account to understand the evolution and differences between different algorithms and models. Table 2.2 shows the topology analysis followed in this review attending to their most differential advances.

Prediction problem	Trajectory / Maneuver	
Prediction target	Ego vehicle /Surrounding vehicles	
Sensors	Camera / LiDAR / Radar	
Variables	Vehicle State	Context
	Position	Road structure
	Speed	Lanes
	Acceleration	Lane marking
	Heading	Appearance
	Yaw rate	
	Width	
	Length	
Interaction	Surrounding Vehicles	
	Free space	
Time domain	Past / Present / Future / Recurrent	

Table 2.2: Key Variables of Analysis in Prediction Works

Prediction problems are clearly divided into two kinds of predictions: trajectories reviewed in subsection 2.2.1 and maneuvers in subsection 2.2.2. However, sometimes trajectories are used as a basis to predict maneuvers and vice versa, and they contribute in both fields.

Another distinction can be made attending to the prediction target. Some works addressed ego predictions problems such as [22], [23], and [24]. These works are valid to develop simple models using ego spatiotemporal variables only but also complex models with a huge amount of information such as ego and surrounding trajectories and precise context information. However, the prediction of ego trajectories or maneuvers is useless, and these models need to be extrapolated to other traffic participants. The information used in these models is usually extremely accurate, but the same information computed from other vehicles is commonly poorer or simply unavailable. This kind of problem happens when the quality and nature of measures changes from the development set to the deployment one. The same problem arises when the non-onboard sensor's information (i.e., infrastructure) is used to develop models.

Sensors represent another differentiation point. Conventional sensors such as cameras and LiDARs generates world representations that need to be processed to extract information. Others, such as radars, provide a kind of discrete representation of the world which can be used directly. Visual or graphic information such as images and even point clouds are commonly used to extract high and low-level information, but they can also be used as raw inputs thanks to the development of GPUs and *Convolutional Neural Networks (CNNs)*.

The mainstream data used in vehicle prediction problems are vehicle state variables such as position, speed, acceleration, orientation, and yaw rate. These variables define the kinematic and dynamic state of a vehicle, and they are useful to understand past, present, and future self-evolution. The vehicle's width and length, together with its state, define the occupation of the road space. Contextual variables are sometimes considered with the aim of focusing predictions. This variable tries to represent in a numeric way the scene. As far as roads are defined by human agreements to use them, they can be described by parameters such as lane marking, the number of lanes, lane width, lane curvature, type of lines, entries, and exits. Context can also be represented as images that contain all previous context variables in a non-numeric way.

Vehicles use to share the road with other traffic agents such as cars, motorcycles, buses, and trucks. Ego actions, and consequently, ego trajectories, are caused and affected by these other traffic participants. The same mechanism takes place in the opposite direction, being the ego vehicle a source of change for other vehicles. The road is an interconnected place where all agents are directly or indirectly affected by the behavior of each other. This is a crucial point to develop vehicle prediction algorithms. These interactions can be taken into consideration by using free space representations or the surrounding vehicles' state.

Time is an essential element for predictions. Predictions must be based on past and/or present information and must describe an element in the future such as positions or maneuvers. The kind of prediction can be considered as a single-point prediction when a model generates some information at a specific and fixed time in the future. Other models can generate a few samples of prediction at different but fixed time horizons; this can be defined as multi-point predictions. A special kind of prediction can be performed with recurrent models. They can iteratively perform a single-point prediction being able to perform virtually unlimited multi-point predictions.

2.2.1 Trajectory Prediction

Trajectory prediction problem addressed the forecasting of one or various future positions of an analyzed vehicle or a group of them. Trajectories denote a set of positions with a corresponding timestamp, but a single predicted position combined with the current position can build a trajectory. Positions are used to define a precise location, sometimes in 2D or 3D reference systems, sometimes in local or global frameworks. Independently of the reference system used, positions are described by numbers. It is not odd to think that positions or trajectories predictions are tackled from numeric approaches.

Almost all of the works analyzed are based on the use of variables that describe the motion history of the vehicles using their state representation in a numeric way [25], [26], [27], [22], [28], [29], [30], [31], [32], [33], [34], [35], and [36]. Only one approach addressed the trajectory prediction problem from a graphical perspective [37] generating predictions directly over images.

The state of the art is reviewed following these three categories: data used as input and how it is structured, type of generated data, and databases used to develop the models.

2.2.1.1 Input Data

Input variables come from simple position sequences to complex road representations. Attending to the nature of the data, it can be classified into three main groups.

- Kinematic and dynamic variables. Variables such as position, speed, acceleration, heading, and yaw rate define the state vector in a detailed manner. Early works made use of this vehicle representation (total or partially) considering only the prediction target by itself [25], [26], [22]. These approaches learn simple physical-based motion models that cannot anticipate any maneuver until it has been explicitly observed in the input sequence. Usually *Kalman Filter (KF)*, *Gaussian Mixture Models (GMM)*, *Artificial Neural Network (ANN)*, and *Recurrent Neural Network (RNN)* are commonly used in these approaches.

- Contextual variables. Next-level features, such as lateral and longitudinal positions, lateral speed and acceleration, or heading error represents the combination of the vehicle state and the lane parameters. This transformation includes lane information indirectly, which allows models to learn road-based vehicle trajectories [30], [32]. Lane-keeping trajectories are denoted by a centered position on the lane; however, lane change trajectories will abandon the central part of the lane and generally end up in the central area of the adjacent lane.

- Interaction variables. Vehicles interactions are even more conditioning than lane or road configurations, but its integration can be considered in many different ways. The main problem adding interactions is that the number of involved vehicles varies.

  A simple method to include interactions was addressed in [32] where the TTC is appended to the vehicle state input. TTC represents in a single number if one vehicle is approaching others and raises the necessity of a lane change or a speed reduction. However, the decision to change to the left or right lane depends on many factors, such as the availability of the adjacent lanes or social agreements (i.e., overtaking is only allowed by one of the lanes).

  A fixed spatial configuration is proposed in [31] to incorporate all the existing vehicles into the algorithm. Ther road space is divided into small and equal areas to define a lattice where the state vector (position and speed) of each possible existing vehicle is incorporated together with the ego vehicle state vector. This approach is limited in the number of vehicles, but divisions are small enough to represent all the possible vehicles. However, this approach aims to model how surrounding vehicles' trajectories are affected by the ego vehicle but do not model interaction between surrounding vehicles.

  In [33] the same approach is followed, dividing the road area into smaller rectangular divisions. The state vector of each vehicle is represented in each corresponding division. The difference arises with the use of a so-called *Convolutional Social Pooling* block. This block learns spatial interdependencies of the existing motion histories and tries to assess how the surrounding vehicle configuration affects the prediction target. In contrast with [31], this approach is centered on the prediction target. However, it can be applied to any other traffic participants, but the availability or the quality of the measures will change. In [36] a simple vehicle-centric structure is proposed to integrate adjacent vehicle interactions at six tentative positions around the centered prediction target.

Other proposals [28], and [29] store road configuration examples generating a knowledge database using trajectory stretches and/or the road occupancy configuration. This approach does not need a fixed representation structure as information can be stored independently of the number of vehicles present in the scene.

Vehicles' state, road configuration, interactions, and context information are clustered under appearance in images. In [37], video sequences are used to generate future vehicle locations at the image reference system by means of *Generative Adversarial Network (GAN).* This approach avoids dealing with the problem to codify or model context or interactions; on the other hand, predictions are limited to the image domain.

2.2.1.2 Output Data

Prediction models are clearly conditioned by road structure and the vehicle configuration. Based on this, different kinds of trajectories can be predicted, attending to the nature of the used algorithms.

Attending to the number of trajectories predicted them could be classified into two categories:

- The first category is the single-vehicle trajectory prediction, where the prediction is focused only on one vehicle and surrounding vehicles act as conditioning factors. These approaches need to repeat the prediction process for each existing vehicle; however, this is the common approach.

- The second category is multi-vehicle trajectory predictions, where all considered vehicles are predicted at the same time. This problem approach was developed in [34].

Attending to the type of trajectories generated they can be classified into two categories:

- Discriminative models. These models generate fixed trajectories. These kinds of trajectories are the most common, and they are generated usually by ANN, KF, RNN, and CNN models.

- Generative models. This kind of model learns probability distribution functions from data. Predicted trajectories are generated according to these probability distribution functions. The benefit of these kinds of models is that uncertainty is intrinsically modeled. In [34] and [37] this kind of trajectories are generated by means of *Conditional Variational Autoencoder (CVAE)* and GAN respectively.

2.2.1.3 Datasets

The databases used to develop the reviewed works are limited to some private custom datasets and two public datasets. Some works such as [25], [22], [31] use their own datasets which are not publiciy available. In these cases all this datasets were recorded from onboard sensors. The main public dataset used for trajectory predictions is the NGSIM dataset, it has been used in a wide of works [27], [30], [32], [33], [34], and [36]. A small minority [28] and [29] made use of the PKU dataset. As it can be observed the development of prediction trajectory systems uses massively the NGSIM dataset. This dataset offers precise and non occluded data from an infraestructure point of view in a highway straight stretch.

Finally, table 2.3 presents in a simple view a comparison between the reviewed works. Note that references to ego are related to works that addressed the ego trajectory prediction problem. In the case of works based on non-onboard sensors (NGSIM), the label *single* is used to denote single trajectories that could be considered equivalent to ego trajectories. Reference to *center* are relative to a vehicle considering all their surrounding vehicles, which are abbreviated with notation *surr*. The prediction type is denoted with the letters D and G for discriminative and generative models, respectively.

Work			Input				Prediction	
Authors	Year	Dataset	Kinematics	Context	Interaction	Model	Type	Target
Hermes et al. [25]	2009	Own	Ego	-	-	ANN-RBF	D	Ego
Ammoun et al. [26]	2009	Own	Ego	-	-	KF	D	Ego
Ranjeet et al. [27]	2010	NGSIM	Single	✓	-	NN	D	Center
Wiest et al. [22]	2012	Own	Ego	-	-	GMM	G	Ego
Houenou et al. [28]	2013	PKU	Ego	-	All surr.	CYRA	D	Ego
Yao et al. [29]	2013	PKU	Ego	-	All surr.	Database	D	Ego
Yoon et al. [30]	2016	NGSIM	Single	-	-	ANN	D	Center
Altché et al. [32]	2017	NGSIM	Single	✓	TTC	LSTM	D	Single
Kim et al. [31]	2017	Own	Ego + surr	✓	36X21 Grid	LSTM	D	Surr
Deo et al. [33]	2018	NGSIM	Center + surr	✓	3X13 Grid	LSTM + CSP	G	Center
Hu et al. [34]	2018	NGSIM	Center + Surr	✓	SIMP	CVAE	G	All
Benterki et al. [36]	2018	NGSIM	Center	✓	3X2 Grid	LSTM-GRU	D	Center
Roy et al. [37]	2019	VISDRONE	Single	✓	Appearance	GAN	G	Single

Table 2.3: Trajectory Prediction State Of the Art Summary

2.2.2 Lane Change Prediction

The definition of vehicle trajectory prediction is precise, to know where the vehicle will be some time into the future; there are no discrepancies about that. However, the lane change prediction can be understood in two ways. The literature refers to a lane change prediction as to the detection of a lane-change maneuver before the center of the vehicle crosses the lane markings. However, the lane change is a maneuver developed over time with a lane crossing event approximately in the middle. A prediction strictly means to state something before it happens. According to this definition, a lane-change maneuver is only predicted if it is stated before it has started. Some works intend to predict lane changes before they have started by labeling a few samples before its beginning. However, the results of these works do not evaluate the performance in terms of *predictions*. We will adopt the literature definition in this document, assuming a prediction as a detection before the lane change event.

Lane change or maneuver prediction problem is very similar to the trajectory prediction problem. The differences are only differences in the output of the problem. While trajectories are mainly numerical problems, maneuver predictions are treated as a classification problem. The basis of both problems are correlated, and it is common to find works that addressed the trajectory prediction problem with a previous maneuver prediction and vice versa. Future trajectories can be better predicted if the maneuver is known in advance, as well as maneuvers can be better predicted if the trajectories are known. Motion representations are the ground of predictions, either trajectories or actions. They can be complemented with context information, wich is mandatory since maneuvers are related to road structure, especially in highway scenarios. Interactions between vehicles are also a determinant factor for the analysis of vehicle maneuvers. A study conducted to analyze the most relevant features to predict lane changes [38] concludes that the lateral offset and the lateral speed w.r.t the lane axis together with the relative speed to the preceding vehicle are the three most relevant features. These three variables are a combination of kinematic variables (position and speed), context (lane structure), and interaction (relative speeds).

Similarly, we will follow a review of the state of the art analyzing input variables, type of generated outputs, and the dataset used to develop these models.

2.2.2.1 Input Data

The input variables represent the information used to classify or predict actions or maneuvers of analyzed vehicles. These can be classified into three main groups attending to its nature:

- Kinematic and dynamic variables. Most of the works that addresed the lane change or maneuver prediction problem do it by using kinematic and dynamic vehicle state variables [39], [40], [41], [38], [42], [43], [44], [45], [46], [47], [33], [48], [49], [50], [51]. There are only two exceptions that are not directly based on this vehicle state representation. In [52] the parameters of the vehicle's bounding box are used to classify the action developed by the vehicle. These parameters can represent somehow the vehicle state.

- Context information. Context represents road structure, and actions or maneuvers are road-based actions. Lane changes are specifically related to the change from one lane to another, so lane level information is mandatory at least to be able to generate action labels and highly recommendable as input. The most common way to include context information is by using kinematic and dynamic vehicle variables at the lane reference system [30], [44], [48], and [50], or include variables such as the distance to the lane markings [38].

- Interaction variables. Interactions arise the need for maneuvers. They can be incorporated in many different ways, from simple ones such as relative speeds [38], to complex scene representations. The most used representation model is by using a fixed vehicle configuration which commonly is a 3x2 grid representing the front and rear vehicles on the prediction target lane and the two adjacent lanes [51], [50], [49], and [33], and a small variation of 3x3 grid such as in [39], and [42]. Special consideration is made in [39] where a 3-agent model is proposed, including ego vehicle, prediction target on an adjacent lane, and the preceding vehicle of the prediction target. This specific configuration is a simplification of the 3x2 or 3x3 configuration for cut-in lane-change maneuvers. Each element of the grid is filled using different variables, usually relative positions and speeds. These representations are vehicle-centric approaches. However, in [53], vehicle positions are used to generate an occupancy map over a road scheme representation. This approach is road-centric, and the number of interacting vehicles is not limited nor their spatial distribution.

2.2.2.2 Output Data

Prediction algorithms can differ in their output in three different ways. One difference is the number of predicted actions. None of the works addressed this problem from a multi-target prediction problem. All the models are focused on a single vehicle to perform predictions. This prediction can be extended to all the traffic participants by repeating the prediction process taking each vehicle as the prediction target.

Attending to the type of predictions, this can be generative or discriminative.

- Discriminative models. These models produce maneuver probabilities evaluating how the input data is similar to each proposed maneuver. This type of output is generated by *Case-Base Reasoning (CBR)*, *Radio Frequency (RF)*, *Support Vector Machine (SVM)*, ANN, or CNN models.

- Generative models. These models produce maneuver probabilities evaluating how the input data looks like data generated from each type of maneuver. This type of output is generated by *Bayesian Network (BN)*, *Dynamic Bayesian Network (DBN)*, GMM, *Gaussian Process Neural Network (GPNN)*, and *Long Short-Term Memory (LSTM)* encoder-decoder models.

Attending to the prediction time horizon, predictions can be classified into two categories:

- Detections. Detections refer to the classification problem. A lane-change maneuver is a time-consuming action that concludes when the vehicle arrives at its destination lane. The lane change event is considered as the point when the center of the vehicle crosses the divisor line between the lanes. Classifying the lane change's ongoing action before the lane change event takes place is also considered a prediction. Most of the works addressed the lane change prediction problem from this perspective.

- Predictions. Prediction term refers to point out the lane change some time in advance with respect to the starting point of the lane-change maneuver. Some works such as [42], [44], [49], and [50] consider the lane change prediction problem in this manner. However, their results are provided in terms of classification accuracy instead of lane change anticipation. Since the complete lane change is labeled as a positive detection, the system's actual performance to predict lane changes in terms of anticipation is masked. A deeper analysis of the results would reveal if these models can predict lane changes before they started.

Attending to the predicted maneuvers them can be classified into two categories:

- Lateral maneuvers. Two choices are possible in direction; left or right lane changes. However, the lack of left and right lane changes represents the third state, which means a lane-keeping status. More complex actions such as cut-in, cut-out, merge, incorporation, or exit are particular cases of left and right lane changes that involve the road configuration and the viewpoint.
- Longitudinal maneuvers. These are defined in [33] as a combination with the lateral maneuvers. The stop maneuver represents a reduction from its original speed. This kind of longitudinal classification is oriented to generate previous knowledge of multimodal trajectory prediction systems.

2.2.2.3 Datasets

The datasets used for the maneuver or lane change prediction problem are wider in variety concerning the trajectory prediction case. The use of public datasets is limited to two of them. Some works such as [30], [33], [48], [50] use of the NGSIM, one more time, this dataset has demonstrated to be a reliable source of data for vehicle prediction algorithms. The second public dataset used was the PKU dataset, and the works based on it [46] and [47] are based on the public and private part of this dataset. Most of the works [39], [40], [41], [38], [42], [44], [53], [49], and [51] are based on private onboard datasets. This effort to create datasets to address the lane change prediction problem from onboard perspective reveals the interest from the automotive industry.

Finally, table 2.4 presents a comparison between the reviewed works. Note that references to ego are related to works that addressed the ego lane change prediction problem. The label *single* denotes an isolated vehicle analysis for works based on NGSIM. Reference to *center* are relative to a vehicle considering all their surrounding vehicles, which are abbreviated with notation *surr*. The prediction type is denoted with the letters D and G for discriminative and generative models. P label differentiates predictive works from purely classification ones.

Work			**Input**				**Prediction**	
Authors	**Year**	**Dataset**	**Kinematics**	**Context**	**Interaction**	**Model**	**Type**	**Target**
Kasper et al. [39]	2012	Own	Ego + surr	✓	3x3 Grid	BN	G	Surr
Graf et al. [40]	2013	Own	Ego + 2 veh	✓	3 Veh model	CBR	D	Surr
Kumar et al. [41]	2013	Own	Surr	✓	-	SVM	D	Surr
Schlechtriemen et al. [38]	2014	Own	Ego + surr	✓	Rel. speed	GMM	G	Surr
Schlechtriemen et al. [42]	2015	Own	Ego	✓	All surr.	RF+GMM	G-P	Ego
Yoon et al. [30]	2016	NGSIM	Single	✓	-	ANN	D	Single
Bahram et al. [44]	2016	Own	Ego + surr	✓	Game Theory	BN	G-P	Surr
Yao et al. [46], [47]	2017	PKU	Ego + surr	-	All surr	SVM	D	Ego
Lee et al. [53]	2017	Own	Ego + surr	✓	BV Grid	CNN	D	Surr
Deo et al. [33]	2018	NGSIM	Center + surr	✓	3X2 Grid	LSTM	G	Center
Deo et al. [48]	2018	NGSIM	Center + surr	✓	3X2 Grid	LSTM CSP	G	Center
Patel et al. [49]	2018	Own	Center + surr	✓	3X2 Grid	SRNN	D-P	Center
Li et al. [50]	2019	NGSIM	Center + surr	✓	3X2 Grid	DBN	G-P	Center
Li et al. [52]	2019	DBNet	Ego + surr	-	Bounding Box	PCA+SVM	D	Ego
Kruger et al. [51]	2019	Own	Ego + surr	✓	3X2 Grid	GPNN	G	Center

Table 2.4: Lane Change Prediction State Of the Art Summary

2.3 Conclusions

Previous sections have introduced existing public databases and several algorithms and methods to predict future vehicle states in two different dimensions, maneuvers, and trajectories. The conclusions extracted from this in detail review are exposed below:

- There are many pubic datasets available for the development of different *Intelligent Transportation System (ITS)*, even vehicle prediction models. However, none of them is thought with this main purpose. There is a lack of datasets to develop specific maneuver prediction and trajectory prediction systems. Maneuver predictions need specific human work labeling actions. Precise labeling, defining actions in time, is critical. Humans can understand the scene and figure out or measure these action boundaries.
- Top-view datasets are massively used to develop predictive systems due to their quality and absence of misinformation. Real-world driving scenarios do not provide this kind of information, and future deployable systems must be developed from onboard sensor databases. Lane change prediction technology is closer to the real-world due to its significant interest by ADAS. For this reason, many works are developed using their own onboard datasets, which are private.
- Most of the reviewed works are limited regarding their input data structure. Some are limited in the number of vehicles, others in vehicle configurations, and others are limited to a collection of experiences.
- Road level information is widely used. The acquisition of this kind of information is hard, and it could not be precise when it is retrieved from onboard sensors.

2.4 Main Contributions

After the review of the state of the art, and considering the discussion presented before, the main contributions of this thesis are:

- Development of the PREVENTION dataset. An onboard sensor dataset with the aim to provide specifically designed data for trajectory and lane change prediction models. This dataset is open and free access to the scientific community.

- A social study has been conducted to evaluate human performance to predict lane changes. This study evaluates the PREVENTION scenes to set the basis for further comparisons between new research works. This study has proved that humans react to ongoing lane changes instead to predict them. This fact proves that there is room for improvement over human driving.

- Two novel CNN based models without input restrictions regarding the number of surrounding vehicles are presented in this thesis. The maneuver prediction approach is based on a CNN model and uses appearance to include context and interactions between traffic participants. The trajectory prediction approach is based on a CNN model and uses vehicle graphic representations to pattern interactions. The trajectory prediction system is able to predict trajectories for all the vehicles at a single prediction operation instead of one-by-one predictions.

Chapter 3

Dataset

This chapter describes the *PREdicion of VEhicle inteNTION (PREVENTION)* dataset. This dataset has been created in the context of this thesis to fulfill identified shortages for the development of vehicle intention and trajectory prediction models. In contrast with many other works, this dataset has been made publicly available. The PREVENTION dataset has been used to conduct a study to evaluate the human performance predicting lane changes and to develop trajectory and maneuver predictive models. The study and the developed models will be detailed in chapters 4 and 5 respectively.

The content of this chapter is structured as follows: section 3.1.1 provides detailed information of the recording platform and the sensor setup, metadata and manual labels are presented in section 3.2, and finally, the dataset details are described in section 3.3.

Conclusions derived from the creation of this dataset are summarized in section 5.3.

3.1 Recording Platform

In this section, all the low-level aspects of the PREVENTION dataset are carefully detailed, including a description of the recording platform, sensor setup, calibration procedures, and time synchronization mechanisms.

The recording platform is an automated Citroën C4 equipped with a sort of sensors and hardware. Figure 3.1 shows the mobile platform with the sensor setup on top of it.

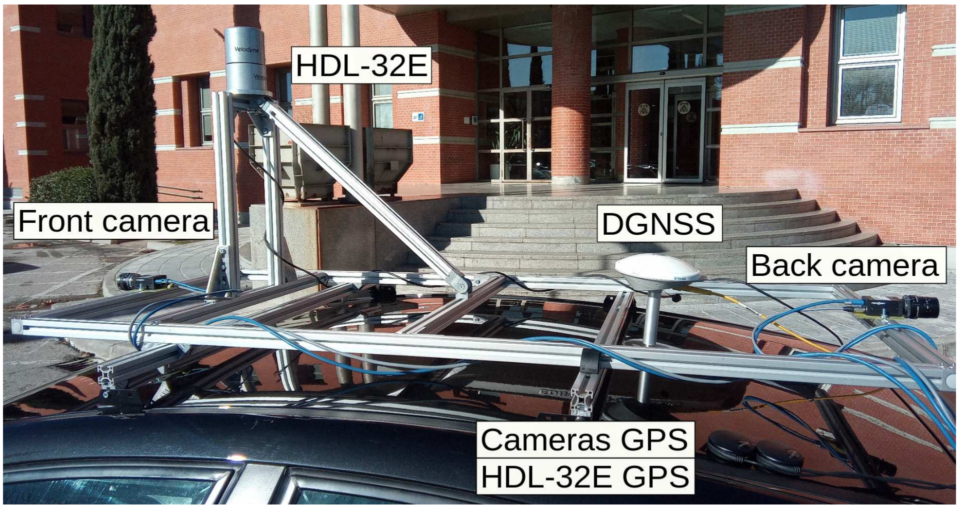


Figure 3.1: Mobile platform and sensors setup.

3.1.1 Sensor Setup

The image acquisition system consists of two Grasshopper3 cameras mounting a 12.5 mm fixed focal length lens. The cameras cover a *Field Of View (FOV)* of 48° in the front and the back. The sensor is a SONY CMOS Bayer array with WUXGA (1920×1200) resolution that can be triggered up to 163Hz.

A Velodyne HDL-32E generates point clouds at a constant rate of 10 Hz. Each cloud is defined by an array of 3D points with 32 vertical and more than 2000 horizontal samples with all-around coverage and +10° to -30° vertical FOV. The detection range of the LiDAR is up to 100 m with an error lower than 2 cm.

Three radars complete the perception system. A Continental ARS308 long-range radar is located centered on the front bumper with a detection range up to 200 m and a FOV up to 56°. Real-time scanning of tracked objects is provided at 16 Hz. Two Continental SRR208 blind-corner radars are installed in both corners of the front bumper with a detection range up to 50 m and a FOV up to 150°. Tracked objects' information is provided at a rate of 33 Hz approx.

A *Differential Global Navigation Satellite System (DGNSS)* Trimble Net R9 Geospatial performs the localization task with RTK capability. Geographical coordinates are generated at 20 Hz with differential corrections received through the 3G/4G network and a Bluetooth connection.

The *Controller Area Network (CAN)* bus of the vehicle is monitored continuously, and many variables such as steering position, braking pressure, throttle position, speed, acceleration, and gear are available and logged.

Finally, an IMU MPU6050 complements the localization task. This low-grade IMU in combination with the CAN bus and the DGNSS data enables better localization and ego-state estimations by means of an *Extended Kalman Filter (EKF)* [54] and a dynamic vehicle model [55].

3.1.2 Data-logging

Data-logging is carried out by three different computers. The central computer is the control computer of the vehicle, which oversees reading data coming from CAN bus, radars, IMU, and DGNSS. This computer generates the ego-vehicle log with the raw data and the time when it was received. A second computer stores the images coming from both cameras. The data flow can reach up to 6 Gbps when the cameras are triggered at their maximum rate. However, in this application, they are triggered at the LiDAR spinning rate, close to 10 Hz generating a data flow of 360 Mbps, which can be supported for continuous operation. The last computer is dedicated to read the LiDAR input and generates the trigger for the cameras. The custom cloud video file with the LiDAR measures is stored on this computer. The log files with the triggering and acknowledgment timestamps of each image are recorded on the same computer.

3.1.3 Ego-Position Estimation

An EKF has been used to fuse the information coming from the DGNSS, the IMU, and the CAN bus of the vehicle. The state vector $\hat{x}$ is estimated using the measure vector z, the non-linear process model f, and the observation model h according to the eqs. described in [56], where E and N are the easting and northing (world) coordinates respectively, ϕ is the heading or forward direction, v is the longitudinal speed and $\dot{v}$ is the longitudinal acceleration.

$$\hat{x} = \begin{bmatrix} E & N & \phi & v & \dot{\phi} & \dot{v} \end{bmatrix}^T \tag{3.1}$$

$$z = \begin{bmatrix} E & N & v & \dot{\phi} & \dot{v} \end{bmatrix}^T \tag{3.2}$$

$$\hat{x}_k = f\left(\hat{x}_{k-1}\right) + w_k, \quad \hat{z} = h\left(\hat{x}_k\right) + v_k \tag{3.3}$$

3.1.4 Extrinsic Sensor Calibration

Spatial calibration between sensors has been carried out in order to enable sensor fusion capabilities. The ego-vehicle reference system is in the middle of the vehicle over the rear axis, where the DGNSS is placed. The x-axis and y-axis match with ego-vehicle's forward and left movement directions. Consequently, the z-axis points up according to a Cartesian right-handed system. Three groups of sensors: radar, camera, and LiDAR have been calibrated to put together all the available information coming from the environment in a common reference system. Figure 3.2 shows all the reference systems defined in the vehicle.

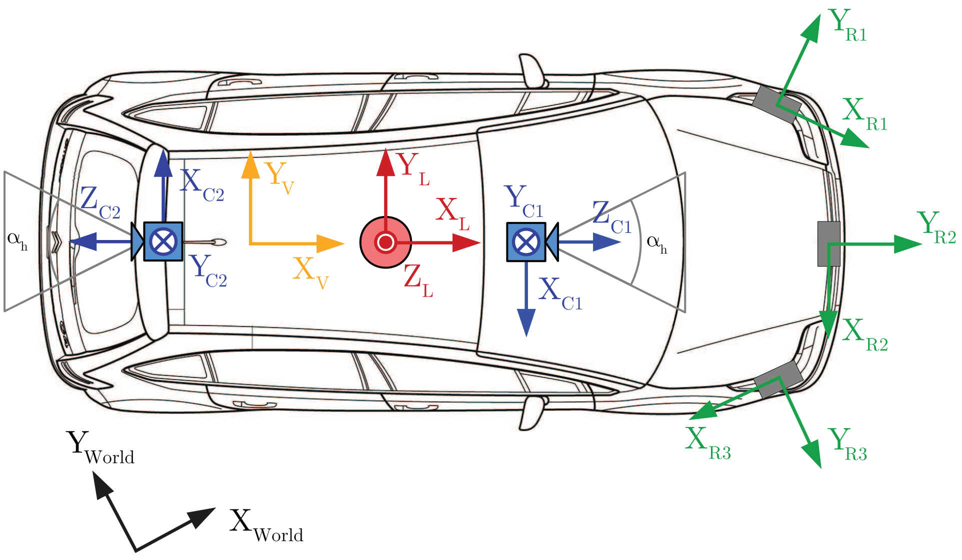


Figure 3.2: Sensor reference systems in the vehicle frame. All the reference systems are Cartesian right-handed systems.

Cameras are intrinsically calibrated and extrinsically w.r.t the LiDAR according to the procedure described in subsection 3.1.5. Radars are extrinsically calibrated w.r.t the vehicle reference system with a procedure based on a Digital Map with known high-sensitive radar elements such as traffic signals and light poles. This procedure is described in subsection 3.1.6. Extrinsic calibration between LiDAR and vehicle reference system is defined by a constant transformation matrix composed with a translation vector. The LiDAR is mounted in a fixed position over a structure built in parallel to the vehicle axes.

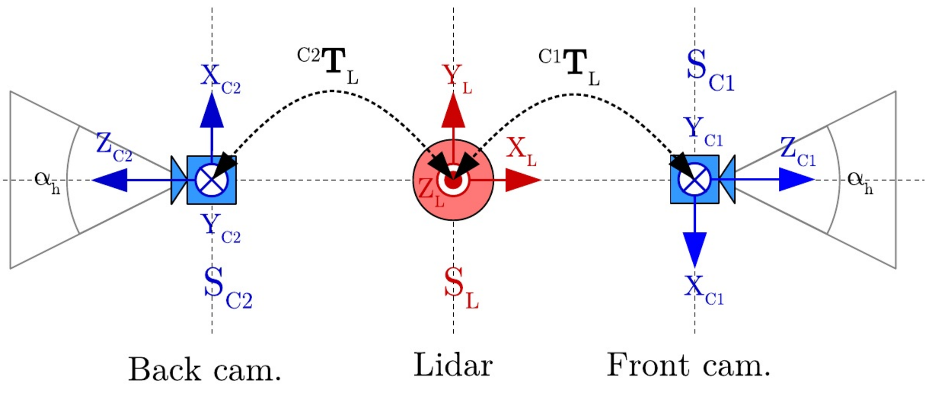


Figure 3.3: LiDAR and cameras reference systems.

3.1.5 Camera and LiDAR Calibration

The extrinsic camera-LiDAR calibration process is based on plane alignment, and it consists of three steps. The first step calibrates the intrinsic parameters of the camera and generates the pattern calibration planes Π_C from the camera point of view. The second step generates manually the same planes Π_L in the LiDAR point cloud. Finally, the last step finds the best extrinsic calibration matrix $^C\mathbf{T}_L$ in a *Singular Value Decomposition (SVD)* fashion and a posterior non-linear optimization algorithm.

The camera-LiDAR calibration process estimates a homogeneous transformation matrix $^C\mathbf{T}_L$, which allows the transformation of points p_L from the LiDAR reference system S_L to points p_C in the camera reference system S_C, and vice versa according to eq. 3.4.

$$p_C =^C \mathbf{T}_L \cdot p_L \tag{3.4}$$

Matrix $^C\mathbf{T}_L$ can be expressed as a rotation matrix $\mathbf{R}_{3\times3}$ and a translation vector $\mathbf{t}_{3\times1}$, as it is shown in eq. 3.5.

$$^C\mathbf{T}_L = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{t}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{3.5}$$

In order to avoid ambiguities regarding the orientation of the calibration pattern planes, its equation has been defined in eq. 3.6. Sub index X represents the sensor point of view, C, or L for the camera and LiDAR. The second sub index refers to the image-cloud pair i.

$$\Pi_{X,i} : ax + by + cz = dd \leq 0||(a, b, c)|| = 1 \tag{3.6}$$

The extrinsic camera-LiDAR calibration process is based on plane alignment, and it consists of three steps. The first step calibrates the intrinsic parameters of the camera and generates the pattern calibration planes Π_C from the camera point of view. The second step generates manually the same planes Π_L in the LiDAR point cloud. Finally, the last step finds the best extrinsic calibration matrix ${}^{C}\mathbf{T}_L$ in an SVD fashion and a posterior non-linear optimization algorithm.

3.1.5.1 Camera Plane Equation Extraction

The camera intrinsic calibration process estimates the intrinsic matrix $\mathbf{K}$ and the lens distortion coefficients. For this purpose, the Matlab® Computer Vision System Toolbox™has been used. Knowing the parameters that model the sensor, the calibration pattern plane Π_C can be found. Figure 3.4 shows a sample used to calibrate the camera, which is one of the planes used for the extrinsic calibration.

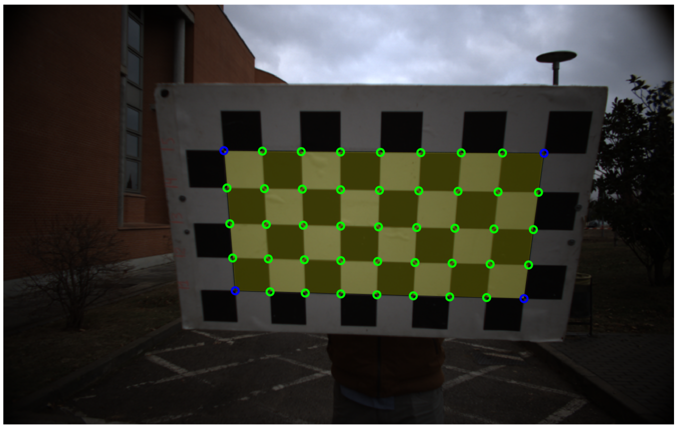


Figure 3.4: Automatic pattern detection in the camera calibration procedure. Green points represent points used to calibrate the camera, and the yellow area represents the plane Π_C

3.1.5.2 LiDAR Plane Equation Extraction

The equation of the calibration pattern, unlike the camera, is not automatically generated for the LiDAR. It is necessary to select the calibration pattern manually in the point cloud for each camera-cloud pair i. This process has three stages:

- Firstly, the corners of the calibration pattern are manually selected; these points are denoted as $P_{M,i}$.
- Secondly, a basic geometric segmentation is performed over the entire cloud to isolate the calibration pattern. The centroid of the calibration pattern $C_{M,i}$ is computed using the manually selected points $P_{M,i}$. An Euclidean-distance based segmentation is performed using the semi-diagonal length of the calibration pattern $d_{CP/2}$ as it is shown in eq. 3.7. Afterward, the manually selected points are used to estimate a plane $\Pi_{M,i}$ in an RMSE fashion. The points p_j in the cloud i are filtered out if they do not satisfy a certain distance threshold d_Π w.r.t. $\Pi_{M,i}$. After this process, the cloud is pruned out, and the remaining points mostly belong to the calibration pattern.

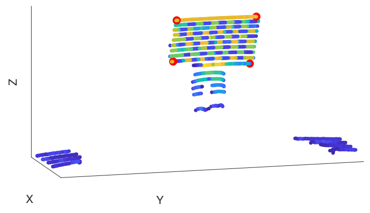
$$d(C_{M,i}, P_j) \leq d_{CP/2} \cap d(\Pi_{M,i}, P_j) \leq d_\Pi \tag{3.7}$$

- Finally, a plane $\Pi_{L,i}$ is fitted using the remaining points p_j. A closed-form robust method has been used to fit the plane (see alg. 1). Firstly, a tentative plane is computed in an RMSE fashion using all the available points. In every iteration $n \leq N$ the distances to the plane d_j are computed and sorted, then the points p_j, which their distances are under the percentile P_α are removed, and a new plane is estimated with the remaining points.

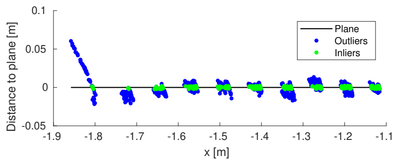
```
Result: $\Pi_L$
$\Pi_L = f(P_J)$;
for $n \leq N$ do
    for $j \leq J$ do
        $d_j = d(p_j, \Pi_L)$;
    end
    $sort(p_j$, based on $d_j)$;
    $P_J = p_1, \cdots, p_{J \cdot P_\alpha}$;
    $\Pi_L = f(P_J)$;
end
```

Algorithm 1: Plane Estimation

Figure 3.5 shows a graphic representation of the calibration pattern plane extraction. Figure 3.5a shows the whole point cloud around the calibration pattern in which the manual corner annotations are represented with red dots. Figure 3.5b represents a plane cross-view of the segmented points around the calibration pattern. Blue and green points represent points that satisfied eq. 3.7. Green points represent inliers after the fitting procedure described in algorithm 1.



(a) LiDAR calibration scene example.



(b) Cross calibration plane view.

Figure 3.5: LiDAR calibration pattern estimation.

3.1.5.3 Camera-LiDAR Transformation Matrix Computation

The final step computes a homogeneous transformation matrix ${}^{C}\mathbf{T}_{L}$ according to the description in eq. 3.5.

The camera-camera extrinsic calibration process uses a set of point pairs as input. However, the corners of the calibration pattern cannot be detected in the LiDAR point cloud, and consequently, the point correspondence cannot be established. An alternative calibration process based on the alignment of the calibration plane pairs has been implemented.

The n plane pairs Π_C and Π_L are used to compute matrices $\boldsymbol{\theta}_C$ and $\boldsymbol{\theta}_L$ and vectors $\boldsymbol{\alpha}_C$ and $\boldsymbol{\alpha}_L$ which represents the normal plane vectors (a, b, c), and the distances to the system reference origin d.

$$\boldsymbol{\theta}_C = \begin{bmatrix} a_1 & \cdots & a_n \\ b_1 & \cdots & b_n \\ c_1 & \cdots & c_n \end{bmatrix}, \quad \boldsymbol{\alpha}_C = \begin{bmatrix} d_1 & \cdots & d_n \end{bmatrix} \tag{3.8}$$

$$\boldsymbol{\theta}_L = \begin{bmatrix} a_1 & \cdots & a_n \\ b_1 & \cdots & b_n \\ c_1 & \cdots & c_n \end{bmatrix}, \quad \boldsymbol{\alpha}_L = \begin{bmatrix} d_1 & \cdots & d_n \end{bmatrix} \tag{3.9}$$

With this notation, the plane equations can be rewritten as:

$$\boldsymbol{\theta}_C \cdot p_C^T = \boldsymbol{\alpha}_C \tag{3.10}$$

$$\boldsymbol{\theta}_L \cdot p_L^T = \boldsymbol{\alpha}_L \tag{3.11}$$

The aligning process firstly estimates the translation vector $\mathbf{t}$. Eq. 3.4 can be rewritten as:

$$p_C = \mathbf{R} \cdot p_L + \mathbf{t} \tag{3.12}$$

The only point which is not affected by the rotation is $\mathbf{0}_L = (0, 0, 0)$. Applying eq. 3.12 to $\mathbf{0}_L$ we have $p_C = \mathbf{t}$. Replacing p_C by $\mathbf{T}$ and p_L by $\mathbf{0}_L$, and subtracting eq. 3.11 to eq. 3.10 the result is:

$$\boldsymbol{\theta}_C \cdot \mathbf{t}^T = (\boldsymbol{\alpha}_C - \boldsymbol{\alpha}_L) \tag{3.13}$$

Eq. 3.13 can be solved now in a *Root Mean Squared Error (RMSE)* fashion to find the translation vector $\mathbf{t}$.

The second step computes the rotation matrix $\mathbf{R}$ in an SVD fashion. The rotation matrix must align the director vectors of the plane pairs, as it is shown in eq. 3.14.

$$\begin{aligned} \boldsymbol{\theta}_C &= \mathbf{R} \cdot \boldsymbol{\theta}_L \\ \boldsymbol{\theta}_L \boldsymbol{\theta}_C^T &= \mathbf{USV}^T \\ \mathbf{R} &= \mathbf{VU}^T \end{aligned} \tag{3.14}$$

Finally, a non-linear optimization algorithm finds the minimum of an unconstrained multivariable function using a derivative-free method [57]. $\mathbf{R}$ is transformed into an unconstrained vector using a quaternion transformation. The translation vector $\mathbf{t}$ is appended to the quaternion. The cost function for the optimization routine is the mean absolute distance for each image-cloud pair from $P_{M,i}$ to the plane $\Pi_{C,i}$.

When $\mathbf{K}$ and ${}^C\mathbf{T}_L$ are computed, p_L can be transformed to S_C and then into S_I according with eq. 3.15

$$\begin{bmatrix} uw \\ vw \\ w \end{bmatrix}_I = \mathbf{K}\,{}^C\mathbf{T}_L^* \begin{bmatrix} x \\ y \\ z \end{bmatrix}_L \tag{3.15}$$

where ${}^{\mathrm{C}}\mathbf{T}_{\mathrm{L}}^*$ is the simplified transformation matrix (last row is removed). Figure 3.6 shows the projection of the color image over the LiDAR scans.

3.1.6 Radar Calibration

The radar calibration procedure tries to estimate the transformation matrix ${}^V\mathbf{T}_R$ that transforms points p_R from the radar reference system

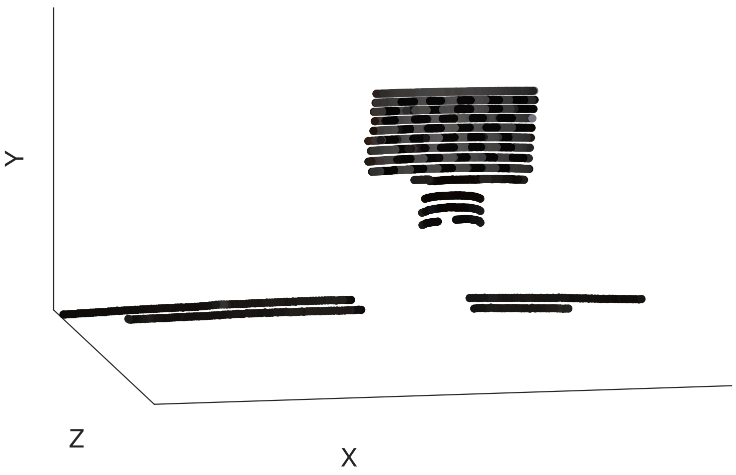


Figure 3.6: Image reprojection over the LiDAR scans after calibration.

S_R to points p_V in the vehicle reference system S_V according to eq. 3.16. In the case of multiple radars, the transformation matrix allows transforming points from one radar to each other in a common frame, thus generating continuous trajectories.

$$p_V =^V \mathbf{T}_R \cdot p_R \tag{3.16}$$

Figure 3.7 shows a representation of the radar distribution and their reference systems together with the vehicle reference system.

Radar-vehicle calibration is like camera-LiDAR calibration, but radars have the particularity to be a 2-dimensional sensor, so the transformation matrix ${}^V\mathbf{T}_R$ can be split into two basic spatial operations, a rotation $\mathbf{R}_{2\times2}$ and a translation $\mathbf{t}_{2\times1}$.

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{2\times2} & \mathbf{t}_{2\times1} \\ \mathbf{0}_{1\times2} & 1 \end{bmatrix} \tag{3.17}$$

The calibration procedure splits the estimation of the transformation matrix in two steps. The first one estimates the rotation matrix, and the second one, the translation vector. The rotation matrix estimation is based on movement properties, but the translation vector needs to be estimated using points correspondence. The challenge here is to find the corresponding pair of points in the vehicle and the radar reference system. The calibration can only be carried out in a well-known environment. In the next subsections, the calibration environment and the radar error propagation are described to finally estimate the rotation and translation parameters.

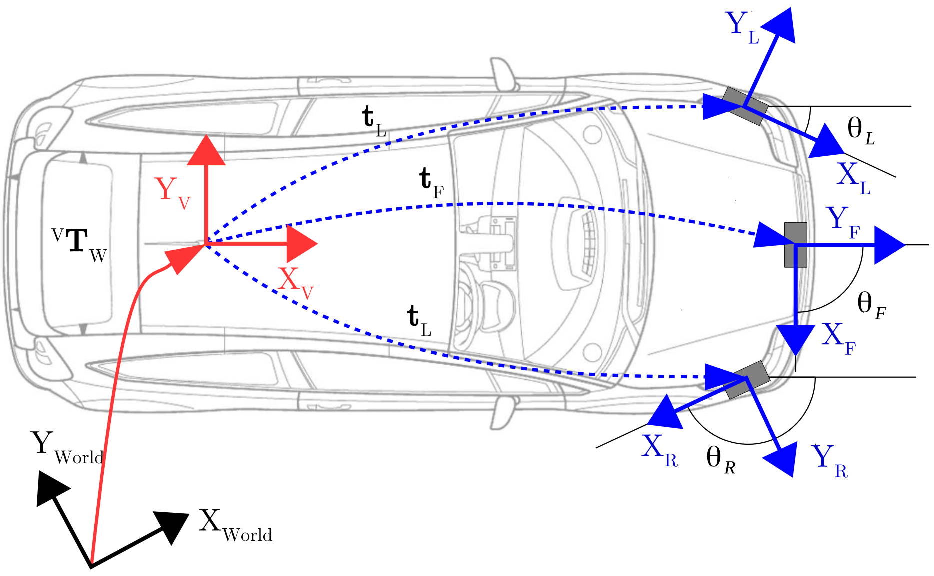


Figure 3.7: Vehicle and Radars reference systems.

3.1.6.1 Calibration Environment

The calibration method exploits the detection of high radar-sensitive structural elements that are static, and their location is fixed in a global reference system. The calibration environment consists of a two ways road with two lanes per way with a roundabout at each end. There is a sidewalk with streetlights and traffic signs on both sides of the road. The streetlights and the traffic signals positions have been used as ground truth in the calibration procedure. Their positions have been measured with a DGNSS with an error lower than 2 cm. The recorded positions are surrounding the vertical pole axis, so the pole axis position is estimated to be the centroid of the measures. The latitude-longitude coordinates are transformed into an easting-northing reference system according to the transverse Mercator projection to get a flat and orthonormal representation reference system. Figure 3.8 shows the position of some of the streetlights and the traffic signals in a high definition digital map representation of the environment.

The position of the calibration elements is transformed from the world reference system S_W to the vehicle (local) reference system $\S_V$. A transformation $^V\mathbf{T}_W$ is performed according to the vehicle state vector $\hat{x}$ defined in 3.1.3.

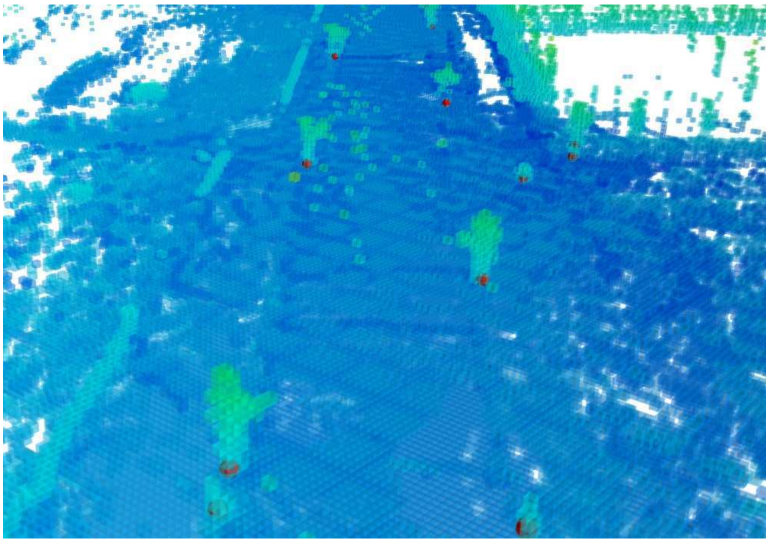


Figure 3.8: Radar calibration targets on a digital map. Red spheres represent the GPS coordinates of the calibration elements.

$$p_V = {}^V\mathbf{T}_W \cdot p_W \tag{3.18}$$

where ${}^V\mathbf{T}_W$ is the inverse of ${}^W\mathbf{T}_V$, and it is defined as:

$${}^W\mathbf{T}_V = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & E \\ \sin(\phi) & \cos(\phi) & N \\ 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

where ϕ, E, and N are ego-vehicle state variables.

3.1.6.2 Radar Measures Error Propagation

The quality of the radar measures varies from close to far detections, from centered to lateral measures. The measurement uncertainty is a parameter to consider in the calibration process. The radar detection accuracy is, in general, provided in a polar reference system just like the measures are. On many occasions, the positions are transformed into an orthonormal reference system, according to eq. 3.20, where ρ is the detection range, α is the direction of the detection and x and y are the orthonormal coordinates. The error of the orthonormal coordinates is propagated as follows:

$$x = \rho\cos(\alpha), \quad y = \rho\sin(\alpha) \tag{3.20}$$

$$\frac{\partial x}{\partial \rho} = \cos(\alpha), \quad \frac{\partial y}{\partial \rho} = \sin(\alpha) \tag{3.21}$$

$$\frac{\partial x}{\partial \alpha} = -\rho \sin(\alpha), \quad \frac{\partial y}{\partial \alpha} = \rho \cos(\alpha) \tag{3.22}$$

Assuming the range error ε_ρ and the direction error ε_α are independent variables the errors ε_x and ε_y can be computed as it is shown in eqs. 3.23 and 3.24.

$$\varepsilon_x = \sqrt{\left(\frac{\partial x}{\partial \rho}\varepsilon_\rho\right)^2 + \left(\frac{\partial x}{\partial \alpha}\varepsilon_\alpha\right)^2} \tag{3.23}$$

$$\varepsilon_y = \sqrt{\left(\frac{\partial y}{\partial \rho}\varepsilon_\rho\right)^2 + \left(\frac{\partial y}{\partial \alpha}\varepsilon_\alpha\right)^2} \tag{3.24}$$

In our case, the position accuracy is limited due to the CAN bus communication protocol. The objects' position has a resolution of 0.1 m, which limits the position error to a minimum value, as expressed in eq. 3.25.

$$\varepsilon_x = \max\{0.1, \varepsilon_x\}, \quad \varepsilon_y = \max\{0.1, \varepsilon_y\} \tag{3.25}$$

When two detections i, j are used to compute the direction of the vector formed by them by using the arctan function, the errors of both detections are involved in the error of the resulted direction. Eqs. from 3.26 to 3.30 describe the computation of the direction error based on the points errors.

$$\theta = \arctan(\Delta y / \Delta x) \tag{3.26}$$

$$\frac{\partial \theta}{\partial \Delta x} = \frac{-1}{\Delta y + \Delta x^2 / \Delta y} \tag{3.27}$$

$$\frac{\partial \theta}{\partial \Delta y} = \frac{1}{\Delta x + \Delta y^2 / \Delta x} \tag{3.28}$$

Assuming $\varepsilon_{\Delta x}$ and $\varepsilon_{\Delta y}$ are independent variables the direction error ε_θ can be computed as follows:

$$\varepsilon_\theta = \sqrt{\left(\frac{\partial \theta}{\partial \Delta x}\varepsilon_{\Delta x}\right)^2 + \left(\frac{\partial \theta}{\partial \Delta y}\varepsilon_{\Delta y}\right)^2} \tag{3.29}$$

$$\varepsilon_{\Delta x} = \varepsilon_{x_i} + \varepsilon_{x_j}, \ \varepsilon_{\Delta y} = \varepsilon_{y_i} + \varepsilon_{y_j} \tag{3.30}$$

The error analysis of each position and the angle formed by two of them set the basis to formulate the rotation and translation estimation in a scoring fashion.

3.1.6.3 Radar Rotation Estimation

The rotation calibration process estimates the radar rotation angle θ_R that aligns the vehicle reference system $\S_V$ and the radar reference system S_R. The matrix $\mathbf{R}$ is a two-dimensional rotation transformation, according to eq. 3.32 and depends only on the parameter θ_R.

$$p_V = \mathbf{R} \cdot p_R \tag{3.31}$$

$$\mathbf{R} = \begin{bmatrix} \cos(\theta_R) & -\sin(\theta_R) \\ \sin(\theta_R) & \cos(\theta_R) \end{bmatrix} \tag{3.32}$$

As far as the relation between the radar detections and the targets is unknown, the traditional calibration approaches based on pairs of points cannot be conducted. An alternative methodology based on the relative movement properties of static objects has been developed. The trajectory described by a static object is seen from the sensor reference system as the opposite of its trajectory. If the mobile reference system (vehicle) performs a trajectory in a straight line, the trajectory of the static object is seen as a straight line in the opposite direction in the sensor reference system. The angle of the trajectory in the sensor reference system reveals the rotation angle θ_R needed to align both reference systems virtually. Figure 3.9 illustrates the trajectory of a static object from the radar point of view, while the vehicle is performing a straightforward trajectory.

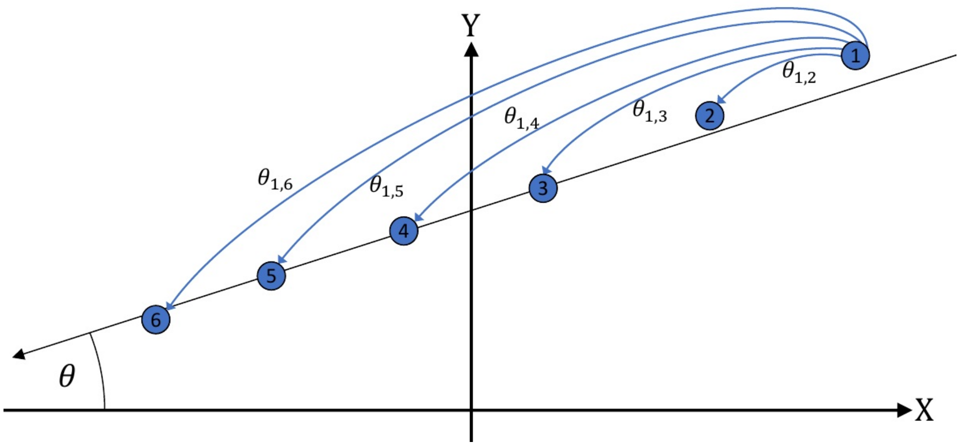


Figure 3.9: Trajectory of a static object from the radar point of view while the vehicle preforms a straightforward trajectory.

Trajectories generated by the objects are a sequence of points in the radar reference system $\S_R$. A single trajectory P is formed by a set of

n points p as it is defined in eq. 3.33, where the sub index represents the time-order of the points.

$$\mathrm{P} = \{p_1, p_2, \ldots, p_n\} \tag{3.33}$$

If the trajectory represents a straight line, the orientation can be easily computed as the arctan function of the extreme points. However, due to the radar detection accuracy, this way is not the best. The end-points of the trajectory are commonly in the limits of the detection area, where the detection error is high. A set of tuples of points TP associated with the trajectory P is defined in eq. 3.34, which generates all the possible combinations of points in the trajectory P in a forward time sense.

$$\mathrm{TP} = \{(p_i, p_j) \mid (p_i, p_j) \in \mathrm{P},\ p_i \neq p_j,\ i < j\} \tag{3.34}$$

The trajectory direction θ is computed for each tuple of points in TP according to eq. 3.26. The error associated to each computed direction ε_θ depends on the individual point errors. The orientation error formula is described in eqs. from 3.26 to 3.30.

Many rotations and their associated errors have been computed for each trajectory P. The trajectories described by many detections must be commonly evaluated to achieve the most reliable value of $\hat{\theta}$. The scoring function proposed to evaluate the set of rotations and errors in a scoring fashion is a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with mean value $\mu = \theta$ and standard deviation $\sigma = \varepsilon_\theta$. The shape of the scoring functions is shown in figure 3.10.

$$s_1(\hat{\theta}, \theta, \varepsilon_\theta) = \mathcal{N}\left(\theta, {\varepsilon_\theta}^2\right) \tag{3.35}$$

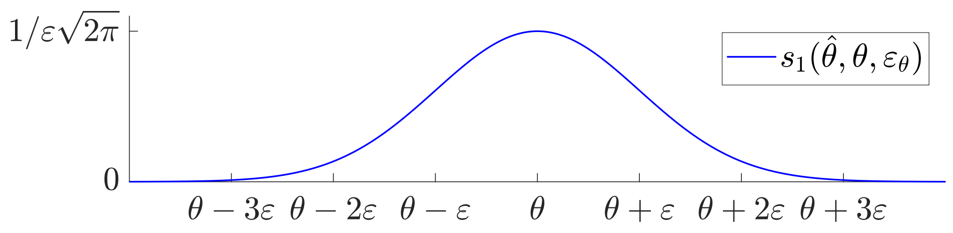


Figure 3.10: Radar rotation scoring function.

The global score is computed as the sum of each score distribution, according to eq. 3.36, where n is the total number of scored angles. The total score distribution is normalized in order to be treated as a probability density function and to provide a confidence interval of the estimation.

$$S_1(\hat{\theta}) = \sum_{i=1}^{n} s_1(\hat{\theta}, \theta_i, \varepsilon_{\theta_i}) \tag{3.36}$$

Finally, the radar rotation angle θ_{R} is minus the $\hat{\theta}$ value with the highest cumulative score according to eq. 3.37 due to the opposite direction of the relative object's movement.

$$-\theta_R = \underset{\hat{\theta}\,\in\,(-2\pi,2\pi]}{\operatorname{argmax}}\ S(\hat{\theta}) \tag{3.37}$$

3.1.6.4 Radar Translation Estimation

Once the rotation between the vehicle and the radar reference systems is known, and they are virtually aligned, the position difference between targets and detections is the translation that is being looked for. Figure 3.11 shows a representation of the relation between the radar measures and the world objects after the alignment process.

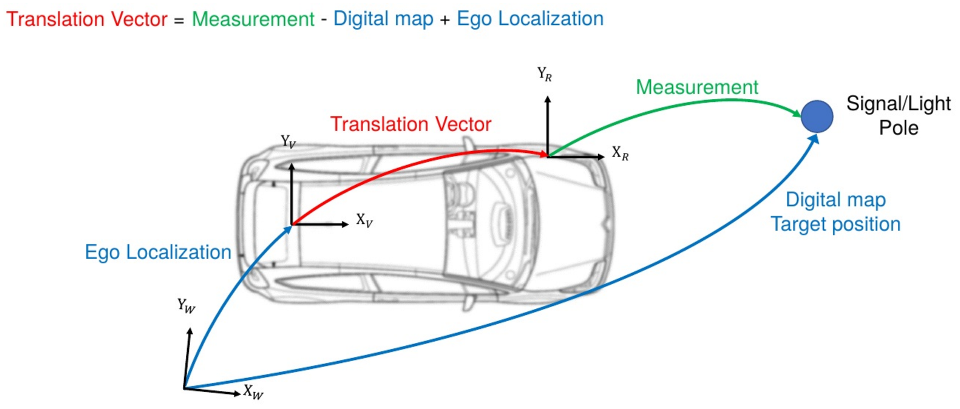

Translation Vector = Measurement - Digital map + Ego Localization
Y_R
Measurement
Signal/Light Pole
Translation Vector
Y_V
X_R
X_V
Digital map Target position
Ego Localization
Y_W
X_W

Figure 3.11: Radar detection and calibration target representation. The world position of the calibration targets can be computed as the sum of the ego-vehicle localization, the radar translation vector, and the radar measure.

Translation estimation has three stages. Firstly, the world points are transformed into the vehicle reference system, according to eq. 3.18. In order to avoid time delays between the radar detection and the ego-estimation, static sequences are used to compute the translation vector. Secondly, the radar detections and their errors are rotated to be aligned with the vehicle reference system, according to eq. 3.31. Finally, the vehicle-radar translation is achieved by scoring each individual detection-target translation.

A two-dimensional translation vector $\mathbf{t}$ defined in eq. 3.38 is needed to transform calibration targets to detections in the common vehicle reference system and vice-versa. The parameters that need to be found are t_x and t_y, which are the translation along the x and the y axis of the vehicle, respectively.

$$\mathbf{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{3.38}$$

The translation vector $\mathbf{t}$ has been limited by using basic information of the vehicle dimensions. The translation vector limit $\mathbf{t}_L$ is defined in equation 3.39 where W and L are the width and length of the vehicle and δ_x and δ_y a safety gap to avoid possible exclusions due to the radar detection errors.

$$\mathbf{t}_\mathrm{L} = \begin{bmatrix} L + \delta_x \\ H + \delta_y \end{bmatrix} \tag{3.39}$$

The set of translation vectors TV is defined in eq. 3.41 as all the combinations of translation vectors $\mathbf{t}_{i,j}$ for each radar detection d_i to each calibration target ct_j. The translation vector is computed as it is shown in eq. 3.40. If $\mathbf{t}_{i,j}$ exceeds $\mathbf{t}_\mathrm{L}$ the pair d_i and ct_j is assumed as a wrong matching and consequently is excluded from the translation estimation process.

$$\mathbf{t}_{i,j} = ct_j - d_i \tag{3.40}$$

$$\mathrm{TV} = \{\mathbf{t}_{i,j} \mid -\mathbf{t}_\mathrm{L} \leq \mathbf{t}_{i,j} \leq \mathbf{t}_\mathrm{L}\} \tag{3.41}$$

The error associated to each translation $\boldsymbol{t}_{i,j}$ is defined as the error vector $\boldsymbol{\varepsilon}_{t_i} = (\varepsilon_{x_i}, \varepsilon_{y_i})$ which is the result of the detection error rotation.

A similar bi-dimensional scoring function has been used to find out the translation vector. The score function is a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with mean value $\mu = \mathbf{t}$ and standard deviation $\sigma = \boldsymbol{\varepsilon}_t$ as it is defined in eq. 3.42. The shape of the scoring functions is shown in figure 3.12.

$$s_2(\hat{\mathbf{t}}, \mathbf{t}, \boldsymbol{\varepsilon}_t) = \mathcal{N}\left(\mathbf{t}, {\boldsymbol{\varepsilon}_t}^2\right) \tag{3.42}$$

Finally, the global translation score is computed as the sum of each single translation score function, as it is shown in eq. 3.43 where n is the total number of valid detection-target translations.

$$S_2\left(\hat{\mathbf{t}}\right) = \sum_{i=1}^{n} s_2\left(\hat{\mathbf{t}}, \mathbf{t}_i, \boldsymbol{\varepsilon}_{t_i}\right) \tag{3.43}$$

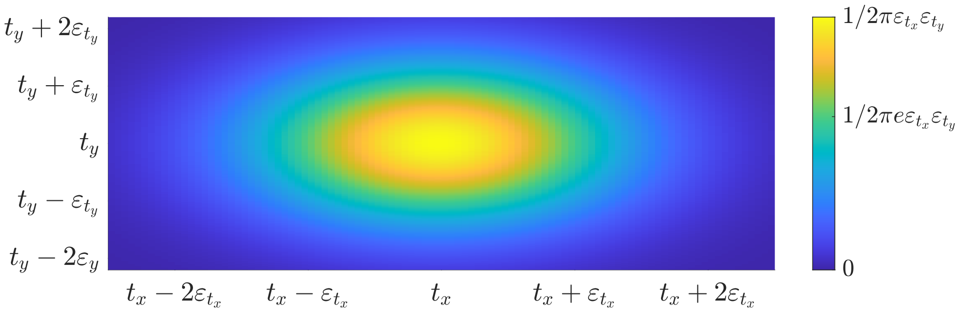


Figure 3.12: Radar translation scoring function.

The estimated radar translation vector $\mathbf{t}$ is achieved finding the translation vector $\hat{\mathbf{t}}$ with the highest S_2 score inside the translation limits $\mathbf{t}_\mathrm{L}$ according to eq. 3.44.

$$\mathbf{t} = \underset{-\mathbf{t}_\mathrm{L} \leq \hat{\mathbf{t}} \leq \mathbf{t}_\mathrm{L}}{\operatorname{argmax}} \; S\left(\hat{\mathbf{t}}\right) \tag{3.44}$$

Figure 3.13 shows an example of the radar detections after the calibration process. Detections of the three radars are commonly represented in the vehicle reference system.

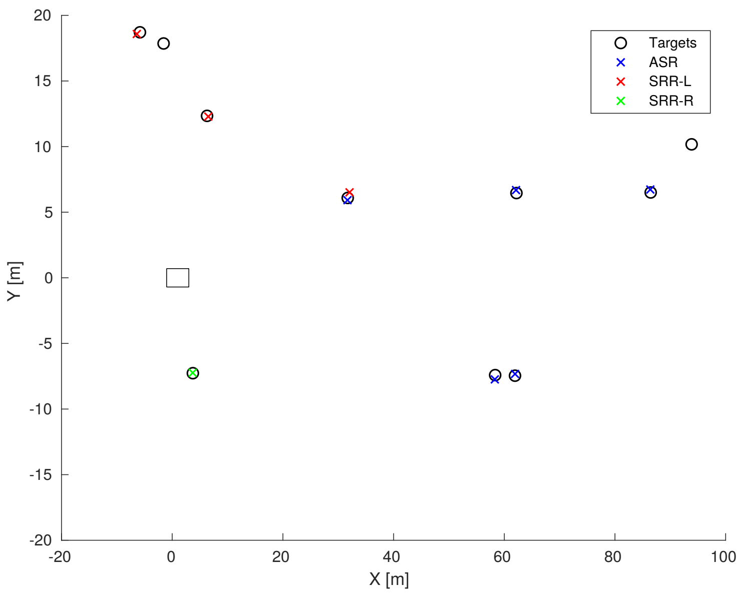


Figure 3.13: Radar reconstruction. Black circles are calibration targets, blue x are ASR308 detections, red x are SRR-L detections, and green x are SRR-R detections.

3.1.7 Time Synchronization

The perception system is composed of sensors of three different types; camera, LiDAR, and radar. They are recorded on three different computers. Spatial-temporal relations between measures are critical when sensor fusion techniques are applied. Focused on this, time synchronization mechanisms are deployed.

Some sensors such a radar, LiDAR, IMU, or CAN bus produce a non-controllable data output. Others, like cameras, are actively triggered, and the data output is known and expected. Two different approaches have been used as a time synchronization mechanism to cover both kinds of data streams.

- Uncontrollable data sources. The clock of the recording computers is synchronized in a common time reference employing a GPS *Pulse Per Second (PPS)* signal and a *Network Time Protocol (NTP)* server. Thus, different recording computers can add a common time stamp to data coming from different sensors at different locations.

- Controllable data sources. Cameras must be externally triggered, and consequently, the data output is actively generated. Cameras are individually triggered when the LiDAR points in the same direction where each camera is pointing to. This guarantees minimum point cloud distortion in the area covered by the cameras. A dedicated computer develops the triggering task and stores the timestamps when the cameras were triggered and when they accepted the trigger signal, which means that the images have been captured.

3.2 Manual Labels and Metadata

This section describes the manual annotations and metadata generated from raw data to enrich the PREVENTION dataset. This effort focuses on providing useful high-level information from raw data. Ground plane coefficients and lane markings are provided as complementary context information. Vehicles have been segmented in the image and temporarily tracked, providing unique IDs for each one. Improved vehicle trajectories are computed fusing image and LiDAR information. Furthermore, finally, manual annotations of observed actions such as lane changes are provided.

3.2.1 Ground Coefficients

The LiDAR points are used to determine the principal plane of the road structure. The plane model is fitted using a fixed region around the ego-vehicle. A cube with 20 meters edges is defined to select the points used to segment the plane. The ground plane coefficients have been computed using a *RANdom SAmple Consensus (RANSAC)* algorithm. The ground model is used to remove ground points and to generate top-view representations.

3.2.2 Vehicle Detections and Tracking

Focusing on vehicle intention and trajectory prediction tasks, vehicle detections are segmented and labeled with a unique ID. The segmentation of the relevant actors is automatically generated using the Detectron framework [58]. To do so, the top-class state-of-the-art Mask-R-CNN [59] model with a ResNet-101 [60] backbone is used as instance segmentation engine. The raw output detections are provided as bounding boxes and contours. Moreover, the temporal integration of the detections is provided. First, the detections with a confidence value lower than 0.5 are filtered out and considered as false positives. Then, a non-maximal suppression algorithm is applied. Figure 3.14 shows an example of automatic vehicle detection after the filtering process.

Finally, detections are temporally associated using a Hungarian Matrix algorithm [61]. The parameter used as the distance between elements is the inverse of the modified *Intersection over Union (IoU)*. Eq. 3.45 presents the modified IoU, where A_1 and A_2 are the evaluated contour areas. The vehicle shape could change along frames being affected by perspective, becoming bigger or smaller. There is no case

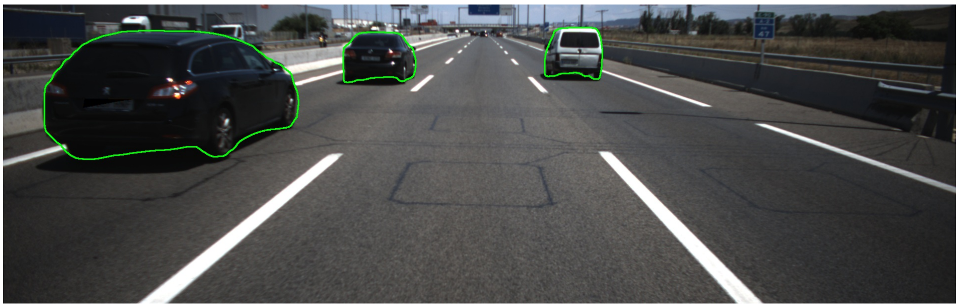


Figure 3.14: Vehicle detection example.

where the intersection of two areas can be greater than the largest of the areas, and this can happen when two shapes of the same object are evaluated at a different time. This formula tries to adapt the IoU to this time-varying representation problem.

$$\text{mIoU} = A_1 \cap A_2 / \min\{A_1, A_2\} \tag{3.45}$$

If the two detections are linked, the new detection adopts the ID of the previous one, creating a sequence of detections for the same ID.

3.2.3 Precise Trajectory Generation

LiDAR has an extremely accurate range detection. However, the angular resolution is limited by its turning rate. For our configuration, spinning at 10 Hz, the LiDAR angular resolution is 0.179°. This number could seem small, but it is amplified by the detection range when a polar to cartesian transformation is applied. In contrast, the camera has a much higher angular resolution. A 1920 pixel width with a 12,5 mm focal length lens, the camera angular resolution is 0.027° . Table 3.1 shows how the angular resolution affects the horizontal error versus the detection distance. The camera is much better than the LiDAR regarding the horizontal resolution and the corresponding lateral error. Despite the camera goodness, it does not provide range measures, so the camera cannot produce 2D or 3D detections. Improved positioning, and consequently, improved trajectories can be generated fusing LiDAR range and horizontal camera detections.

An image point (u, v) cannot be transformed directly into the 3D camera reference system. The pin-hole camera model generates a 3D straight line equation for each given point (u, v) in the image plane, according to eq. 3.15. This means that multiple solutions are possible

Target Distance	**10**	**20**	**50**	**100**	[m]
Camera Resolution	2.2	4.4	10.9	21.9	[mm]
HDL-32E Resolution	15.6	31.2	78.1	156.2	[mm]

Table 3.1: Horizontal Positioning Resolution.

but knowing one of the coordinates of the point x_C, y_C, or z_C, the others are fixed. Making use of the camera-LiDAR extrinsic calibration described in 3.1.5, the LiDAR detections can be transformed into the camera reference system and be used to solve the indeterminacy problem. The key is to select the most accurate measure from each sensor as it was exposed above. The x and y coordinates are generated using the precise grid representation of the camera. The z coordinate, also called depth, comes from the LiDAR. The z camera coordinate is mostly related to the x coordinate of the LiDAR according to the sensor configuration (see figure 3.3). Eq. 3.46 shows the transformation from polar to cartesian LiDAR coordinates where ρ is the range, ϕ is the elevation angle, and θ is the azimuth angle. Note that lateral resolution affects y_L mostly because of the $\sin(\theta)$ term, however, x_L is affected by the $\cos(\theta)$ term which is much less sensitive to the change in the area where θ resolution and ρ takes relevance. This happens at far distances in the front and the back of the vehicle.

$$\begin{aligned} x_L &= \rho\cos(\phi)\cos(\theta) \\ y_L &= \rho\cos(\phi)\sin(\theta) \\ z_L &= \rho\sin(\phi) \end{aligned} \tag{3.46}$$

Combining the camera and the LiDAR, the measurement range is extended virtually to 100 meters. However, due to the LiDAR layers distribution, the maximum detection range over the ground plane is reduced to 75 m. Figure 3.15 shows the relevant LiDAR layer distribution regarding this limitation. The first tilted down layer has an elevation of $-1.33°$, this layer intersects with the ground plane approximately at 85 m. However, the vehicles are not in contact with the road on the front or back bumper. Assuming a maximum clearance of 0.3 m w.r.t the ground plane, the maximum detection distance decreases to 72 m. The horizontal layer can detect vehicles with heights above the LiDAR location (2 meters), such as vans, trucks, and buses, and the detection range is not reduced.

For a precise trajectory reconstruction fusing image and LiDAR data, two things are needed: an image coordinate (u, v) and the cor-

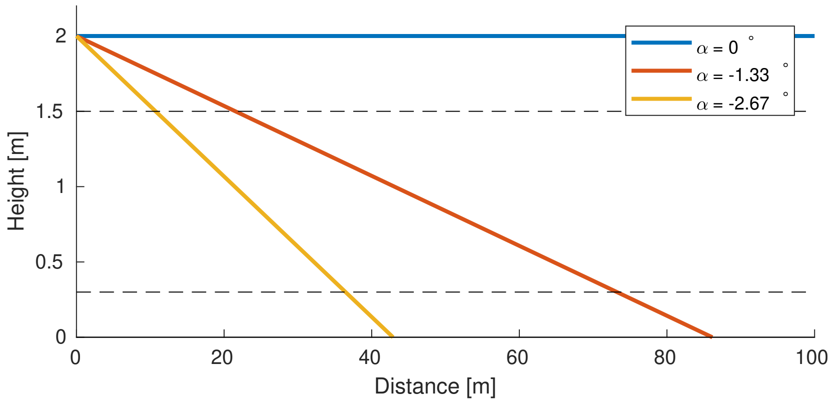


Figure 3.15: LiDAR detection range.

responding depth. The depth is provided as a LiDAR measure with a very precise x_L coordinate.

3.2.3.1 Image Labeling

Manual labeling of all the vehicles in all the frames is a monotonous time-consuming task. A state-of-the-art Median-Flow tracker algorithm has been used in order to semi-automate the labeling process. Initially, the tracker is set up with a *Region of Interest (ROI)* containing the desired key point to be tracked and consecutively updated. This supervised tracking reduces the labeling time, and the labeler can always change the points and reinitialize the tracking if the performance is not good enough. It is important to note that the tracked key point will be used to determine the vehicle location, so it must be laterally centered on the vehicle.

Figure 3.16 shows an example in which the vehicle center is labeled a single time, and it is correctly tracked for more than 30 frames without modifications. The green rectangle represents the area tracked by the algorithm, and the red plus symbol is the ground truth for the corresponding key point. Commonly the band logo is used as the key point for the tracking because it is commonly placed in the middle of the front and rear bumper. Other times the license plate is used as the key point as well.

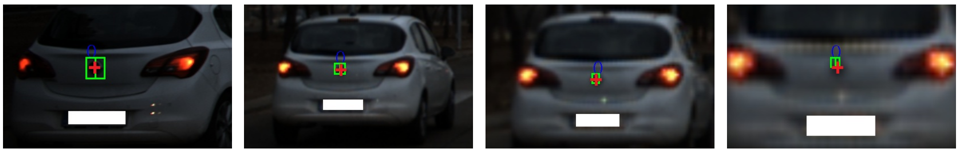


Figure 3.16: Vehicle key point tracking using Median-Flow tracking algorithm. From left to right, tracker initialization and key points after 1, 2, and 3 seconds of tracking. Different zoom levels are applied in each image, 1, 2, 4, and 8 zoom factor respectively.

3.2.3.2 Depth Estimation

Once the vehicles or a part of them are labeled, it is necessary to assign a depth value or z coordinate to this region. The z coordinate is computed using the LiDAR point cloud. This process has four steps:

- First, the LiDAR points under 0.2 m from the ground plane are removed for better depth estimation.
- Second, the LiDAR point cloud is projected over the image plane.
- Third, the points that fit inside a vehicle detection are associated with that vehicle.
- Finally, the closet point of each vehicle is used as a depth estimation for that vehicle detection.

The vehicle's 3D coordinates can be computed in the camera reference system using the image coordinate and the depth value. Figure 3.17 shows an example of the trajectory reconstruction using the Median-Flow tracking algorithm and the depth estimation. This representation shows two trajectories, one in the back of the vehicle (with negative longitudinal values) and other in the front of the vehicle (with positive longitudinal values). The red trajectory was generated selecting the points manually in the point cloud. The low lateral resolution of the LiDAR produces lateral jitter coordinates. The green trajectory uses the manually selected points in the LiDAR as depth value and manual annotations on the image. This trajectory is the best that can be achieved fusing LiDAR and camera data but with a high manual effort. Note that even using jitter LiDAR points, the lateral coordinate does not jitter because of the camera lateral resolution. The blue trajectory is computed using the semi-automatic procedure described above. This method outcomes LiDAR trajectories and reduces manual labeling effort.

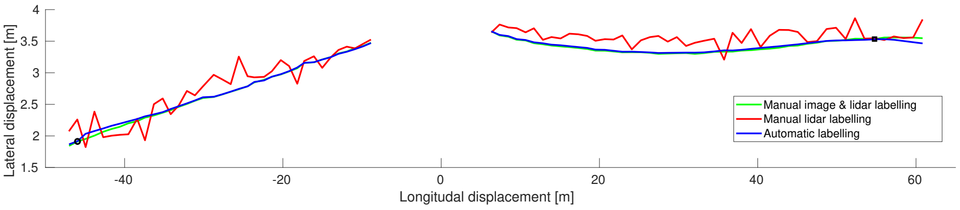


Figure 3.17: Camera and LiDAR-based trajectory reconstruction.

3.2.4 Maneuver Registration

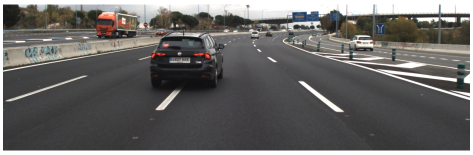
The essential feature when addressing the lane change and trajectory prediction problem is the lane change annotations. These annotations record the events and actions that take place on the road. Events or actions can be classified into two categories attending to its duration. They can be instant actions, such as the blinker's trigger, the activation of the braking light, or the moment when the vehicle crosses the divisor line between two lanes, or actions developed over the time, such as lane changes. Regarding the type of action or event, the most common in highway scenarios are the lane changes. A lane change can be a left or a right lane change. This set can become more complicated if the ego-lane is included in the classification. There could be cut-in and cut-out maneuvers combined with left or right lane changes. Merging actions from entry ramps or leaving actions to exit ramps can be also combined with the primary type of lane changes.

All the lane changes and relevant observed actions have been carefully labeled. The lane-change maneuvers are meticulously reviewed; images are visualized in forward and backward sense to establish precisely the starting frame of the lane changes, the frame when the vehicle crosses the divisor line between lanes, and the moment when the action has finished. The following criterion is used to set the beginning of the lane change: the earlier from the beginning of the lateral movement or the blinker's activation. Note that the blinker's activation could take place a few seconds before the vehicle begins its maneuver.

Figure 3.18 shows an example of a lane change labeling. Figure 3.18a is the first frame when the turn indicator is seen activated. This frame sets the beginning of the lane-change maneuver. Figure 3.18b shows the first frame when the lateral movement of the vehicle is observed, 24 frames later of the blinker activation. Figure 3.18c shows the frame corresponding to the *Lane Change Event (LCE)*. Finally, figure 3.18d shows the end of the lane change, when the lateral movement of the vehicle has ended up.

(a) Activation of turn indicator.

(b) Starting of lateral displacement.


(c) Lane change event.

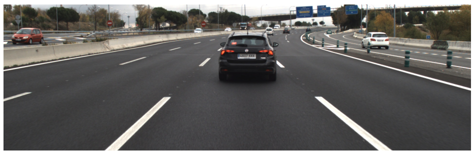

(d) End of lane-change maneuver.

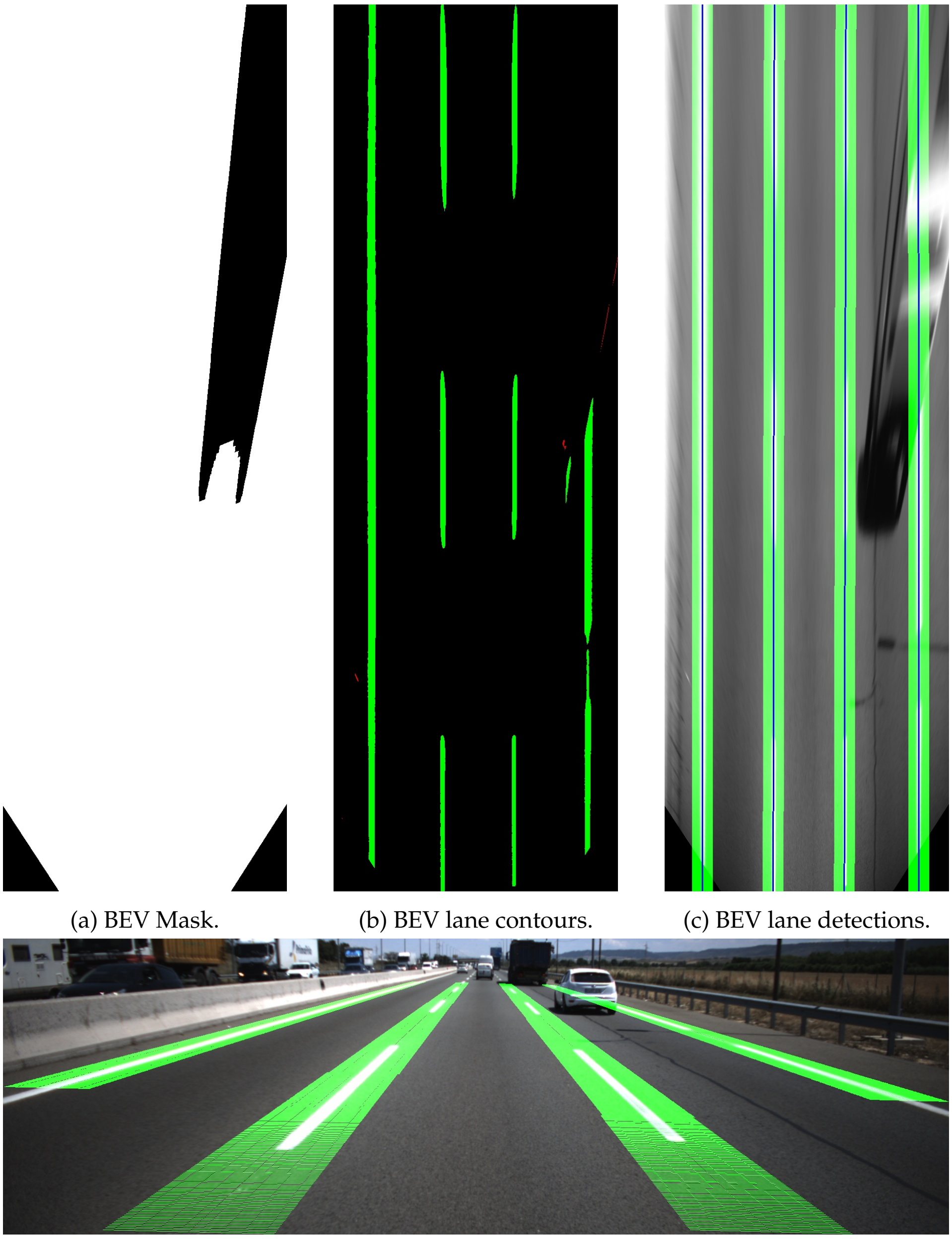
Figure 3.18: Vehicle lane change sequence.

3.2.5 Lane Detection

The road configuration is of paramount importance to understand the context situation. The relative positioning of surrounding vehicles w.r.t. the road lanes enhances the scene understanding. A custom lane detection system [62] detects and tracks each road lane marking. The images are analyzed into a BEV perspective. The original image is converted to a BEV representation according to the ground plane coefficients. A BEV mask is also created including the vehicle detections (see figure 3.19a) to support the lane extraction process. A median filter operation processes the original input image. The median filter image is subtracted to the original one, and the result is thresholded using an adaptive threshold. The contours generated after this process are shown in figure 3.19b. Finally, the contours are filtered by size and shape to remove some noise. The remaining contours are used to fit the lane models. Figure 3.19c shows how easy the lane models fit over the lane markings in the BEV representation. Figure 3.19d shows the same lane models over the perspective image. It can be observed that the vehicle occlusion does not alter the lane estimation in the right-most lane.

Lanes are modeled as a 2^{nd} order polynomial according to eq. 3.47 where c_0 is the lateral distance to the line, c_1 is the angular misalignment and c_2 represents the lane curvature.

$$y = c_2 x^2 + c_1 x + c_0 \tag{3.47}$$



(a) BEV Mask.

(b) BEV lane contours.

(c) BEV lane detections.

(d) Lane detection on the original image.

Figure 3.19: BEV lane detection process.

3.3 Dataset Structure

The PREVENTION dataset contains thousands of examples recorded on real driving conditions and different environments to provide specialized data for the prediction of vehicle intentions and trajectories. The PREVENTION dataset presents the following characteristics:

- Data from 6 sensors of different nature (laser, radar, and vision) are provided, contributing to redundancy and fault-tolerant development. Measurements from the 6 sensors are time-synced and cross-calibrated [63] [64].
- Surrounding data are provided in a range of at least 80 meters around the ego-vehicle (up to 200 meters in the frontal area). This allows for developing a safety area around the ego-vehicle in which all vehicles entering or leaving such area are carefully located and tracked to predict their most likely trajectories accurately.
- Positions of all vehicles around the ego-vehicle are accurately labeled in a semi-automatic process [63] and made available together with their respective vehicle IDs. The fusion between the appropriate sensors is carried out in order to obtain as much an accurate positioning as possible, both in lateral and longitudinal dimensions.
- Road lane markings are included in the dataset, providing the relative positioning of all vehicles on the road (lateral positioning and orientation), including the ego-vehicle and the number and type of road lanes present on the road. This information is essential for enhancing road scene understanding and for providing contextual framing.
- Ground plane coefficients are provided to enable road-based transformations such as BEV.
- PREVENTION dataset also offers specific types of critical maneuvers that are of interest for prediction purposes. Maneuvers of surrounding vehicles are carefully labeled to identify critical situations such as overtaking, merging, and lane-change maneuvers.

In this section, the structure of the PREVENTION dataset is detailed. This includes a description of the recording area, a list of the provided data, and how it is structured.

3.3.1 Driving Environments and Driving Style

PREVENTION dataset contains both urban and highway scenarios, but it is mainly oriented to predicting intentions and trajectories on highway environments. Urban areas are limited to the campus area and some residential areas before entering the highway. Three different person drove the car to generate the data. Drivers were instructed to arrive at the destination following the traffic rules. Drivers used the cruise control at their will. A total length of 6 hours and more than 500 km were recorded in five different days and three different areas. Table 3.3 summarizes the recording details. The A2 and A3 are both three-lane highway areas with straight stretches mostly. The M-50 and M-40 are the outer and middle rings around the city of Madrid, respectively. These rings have three lanes or more. The recordings were made during the central hours of the day to avoid heavy traffic. However, traffic jams and congested traffic can be found in the dataset. Some of the records cover the same driven areas. By doing so, different behaviors and interactions can be observed at the same location with different points of view. This dataset is not created for localization algorithms, but it can be used to develop them with multiple records of the same areas.

Table 3.2: Dataset Recording Main Features

Record #	Drives	Area	Date	Length	Distance
1	A2	1	21st Jun	18 min	47 km
2	A2, M50, A3	2	19th Jul	59 min	83 km
3	A2, M50, A3	2	24th Jul	57 min	86 km
4	A2, M-40	3	18th Oct	108 min	149 km
5	A2, M-40	3	22nd Nov	114 min	175 km
Total	–	–	**356 min**	**540 km**	

3.3.2 Data Access

The dataset is publicly accessible at http://prevention-dataset.uah.es. For simplicity, all the files of each drive have been packaged in two files due to the large size of the raw data. The post-processed data and the labels can be downloaded independently.

3.3.3 Data-format

The database format is structured as follows, where X, Y, N, and M are used to define the record number, the drive order in each record, the camera identifier, and the radar identifier. Figure 3.20 shows the directory tree of a record.

```
DATABASE
└── RecordX
    ├── DriveY
    │   ├── detection_cameraN
    │   │   ├── detections.txt
    │   │   ├── detections_filtered.txt
    │   │   ├── detections_tracked.txt
    │   │   ├── labels.txt
    │   │   ├── lanes.txt
    │   │   ├── lane_changes.txt
    │   │   └── trajectories.txt
    │   ├── detection_cloud
    │   │   └── ground_coefficients.txt
    │   ├── detection_radar
    │   │   └── detections_radarM.txt
    │   ├── logs
    │   │   ├── log_ego-vehicle.txt
    │   │   └── log_cameraN.txt
    │   ├── pcap_radar.pcapng
    │   ├── video_cameraN.raw
    │   └── video_velodyne.bin
    ├── cameraN_extrinsic_calibration.dat
    ├── cameraN_intrinsic_calibration.dat
    ├── radarM_extrinsic_calibration.dat
    └── velodyne_extrinsic_calibration.dat
```

Figure 3.20: Dataset Format.

Extrinsic and intrinsic sensor calibration files are provided for each record. These calibration files enable the transformation of data between the different sensor reference systems. Two camera-LiDAR, one LiDAR-vehicle, and three radar-vehicle transformation matrices are provided for each record. Transforming points from radar to the vehicle reference system assumes that radar detections are at $z = 0$.

Each drive contains raw sensor inputs. Two raw camera videos, a LiDAR cloud video, and radars data transmissions are provided for each drive. The original image size has been reduced to 1920×600, removing the top and bottom bands 1920×300 with irrelevant information to keep the file size smaller. Each image is stored as 1920×600 bytes in a *BayerBG* pattern codification. Camera calibration files have been modified to work properly with the new camera resolution. LiDAR data is codified in a custom cloud video data format. Each *frame* has a fixed size and stores the cloud number, the GPS triggering time, the 3D information, and the returned intensity. The cloud video structure definition is provided together with the cloud video file. Radar data is provided in a parsed format in a text file due to the complex parsing procedure.

Folder `detection_cameraN` contains different information extracted directly from or related to images:

- `detections.txt` provides automatically generated bounding boxes and contours of objects in the scene.
- `detections_filtered.txt` are the same detections with a minimum confidence value of 0.5 and non-maximal suppression.
- `detections_tracked.txt` is the result of a temporal tracking of the filtered detections, assigning a unique id to each object along the frames. Data is stored as a sequence of [$frame$, id, $class$, x_i, y_i, x_f, y_f, $conf$, n] values followed by n tuples of x, y coordinates that represent the contour. Values x_i, y_i and x_f, y_f represents the top left and right bottom coordinates of the detection bounding box. There are more than 4 million detections, including vehicles and pedestrians. More of 3.5 million of these detections are cars and 0.5 are trucks. Pedestrian, motorcycle, bicycle, and bus classes are a minority. This is explained because most of the recording time was on highways,where pedestrians and bicycles seldom appear.
- `labels.txt` is a sequence of [$frame$, id, x, y, $width$, $height$] values for each manual annotation that describes a key point laterally centered on the vehicle. These annotations are used to compute the vehicle improved trajectories. There is a total of 1.3 million of manual annotations that identify unequivocally each vehicle with a unique id and the image coordinates that represent its positions along the frames.

- `lanes.txt` is a sequence of [$frame$, n, c_0, c_1, c_2] values that represents the n lane lines in the scene as a 2nd order polynomial (Eq. 3.47).

- `lane_change.txt` is a list of seven values [id, $type$, $f0$, ff, $val1$, $val2$, $val3$]. The first value id is the id of the vehicle which the action is referred to. Note that all the registered annotations are object oriented. The second value $type$ encodes the type of registered event or action. The third value $f0$ is used to establish the event frame for instant actions or the beginning of a time-lapse action and ff represents the end of the time-lapse action. Variable $type$ could be *left* (1), *right* (2) for lane changes, *hazardous* (3) such as stopped vehicles on the shoulder, or emergency vehicle overtaking, and *zebra crossing* (4) for pedestrian oriented action. Variables from $val1$ to $val3$ are used to provid extra information of lane-change maneuvers. The LCE, is stored in $val1$, and the use of the blinker in $val2$. Variable $val3$ adds information to the lane-change maneuver and can take values *cut-in* (1), *cut-out* (2) when the vehicle arrives to or leaves the ego-lane or *none* (0) otherwise. More than 900 lane changes have been manually labeled. Table 3.3 shows a summary of the basic lane change types *left* or *right* for each record.

Table 3.3: Lane Change Statistics

Record #	**1**	**2**	**3**	**4**	**5**	**Total**
Left	22	36	46	139	170	413
Right	51	48	47	175	178	499
Avg. frames per LC	40.6					
Avg. LC length	3.76 s					

- `trajectories.txt` is a sequence of [$frame$, id, x_c, y_c, z_c, x_l, y_l, z_l] values that represents the 3D position of a vehicle in the camera and the LiDAR reference system. There are more than 3000 trajectories composed from more than 1.3 million samples.

Folder `detection_cloud` holds the file `ground_coefficients.txt` where the coefficients of the principal plane are stored as a sequence of [$frame$, A, B, C, D] values expressed in m^{-1} of a scalar plane equation (eq. 3.48) and relative to the LiDAR reference system.

$$Ax + By + Cz + D = 0 \tag{3.48}$$

File `detections_radarM.txt` contains parsed radar data. Wide-range (1 and 3) and long-range (2) radars are different models, thus provide mostly common but also specific information. Each radar sends a transmission with a snapshot of 25 or 40 objects for the wide-range and the long-range respectively.

- Object #: number of the object in the transmission.
- ID: unique id (wide-range only).
- y, x: longitudinal and lateral distances in meters.
- v_y, v_x: longitudinal and lateral speeds in m/s.
- RCS: radar cross section.
- LT/PoE: lifetime in seconds or probability of existence (wide-range/long-range).
- t: reception time in microseconds.

Simple C/C++ and MATLAB examples are provided in the dataset website to load, use, and show the available information. As an example of this visual information figure 3.21 shows a sequence of integrated information of point clouds, images, radar, and automatic and manual annotations. The left column shows the front camera's acquisitions, and the left column, the acquisition of the back camera. Both rows show a forward time sequence from left to right. Figure 3.22 is a collage of some of the events that could be found in the dataset, such as cut-in and cut-out maneuvers, lane changes to both sides, including entrance and exit ramps, hazardous situations, and pedestrian zebra crossings.

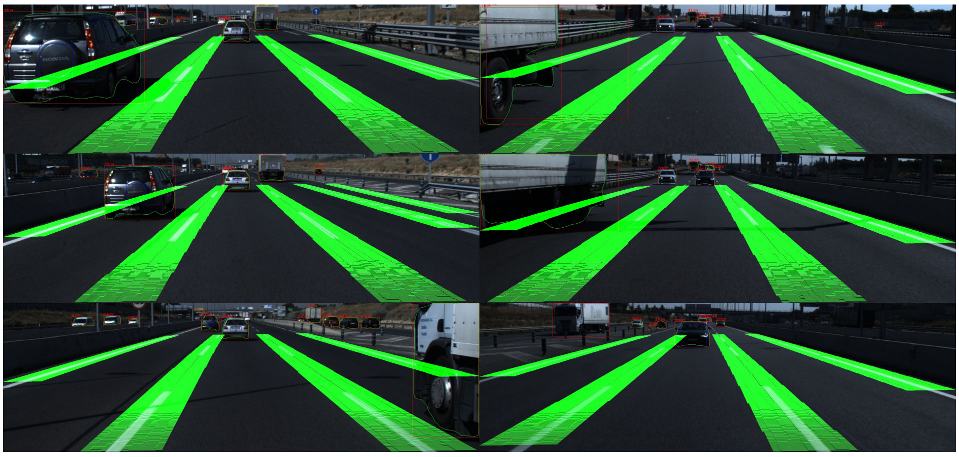


Figure 3.21: Sequence of vehicle detections and tracking. Left images are front, and right images are rear camera records. From top to bottom, samples are progressing in time.

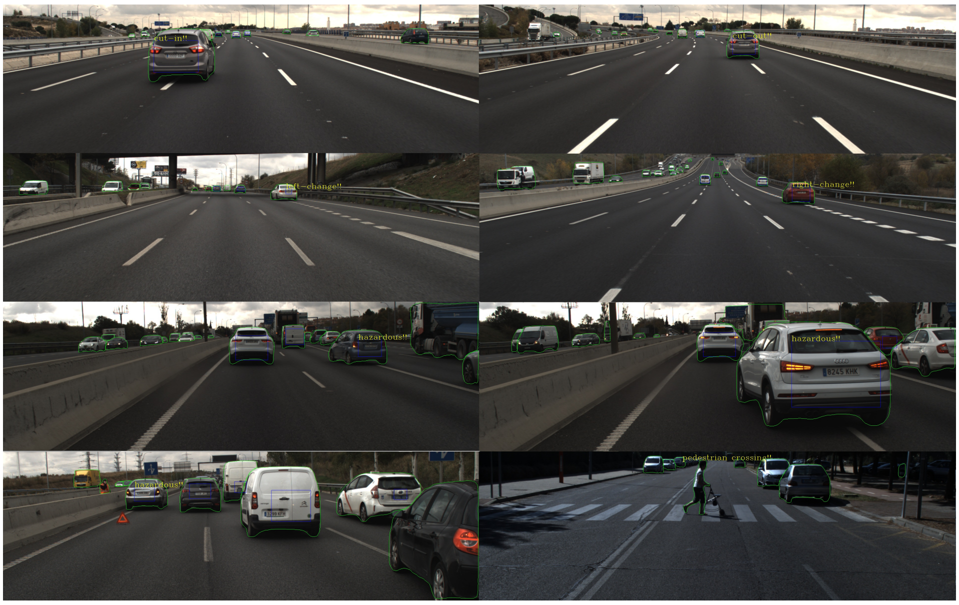


Figure 3.22: Example of different occurrences in the dataset. From left to right and top to bottom, cut-in, cut-out, left-lane change, right-lane change, three hazardous events, and pedestrian crossing.

3.4 Conclusions

This chapter describes the creation of the PREVENTION dataset to fulfill identified shortages in the publicly available datasets. With this objective, after the creation of the PREVENTION dataset, the following conclusions can be extracted:

- The PREVENTION dataset contains more than six hours of naturalistic highway driving with sensor redundancy. Different raw data sources such as cameras, LiDAR, and radars are provided as well as manual annotations, automatic detections, and high-level information. This dataset contains more than 4 million vehicle detections, 3000 trajectories, and 900 lane changes. The PREVENTION dataset can be widely used in different research contexts, focusing on intention and vehicle trajectory prediction.
- Sensor cross-calibration methods have been developed and implemented to enable sensor fusion and fault-tolerant techniques.
- Manual labeling effort has been focused on trajectories generation and maneuver annotation. Because of this, the lateral movement of the vehicles has been manually labeled to provide enhanced vehicle trajectories fusing camera and LiDAR information. A special effort has been made to label all the actions that take place in the scene in which the ego-vehicle can be involved/affected or not, such as the lane changes and potential hazards.
- Contextual information such as lanes, ground planes, or vehicle tracking is extracted from raw data and provided ready to use.
- This information has been made publicly available to the scientific community at prevention-dataset.uah.es.

Chapter 4

Human Ability for Maneuver Prediction

In this chapter, the ability of humans to predict lane changes is evaluated using scenes recorded in the PREVENTION dataset. This study has the goal to state if humans can predict lanes changes, how precise they are, and how much they can anticipate them. Moreover, the result of this study sets a baseline to be compared with future developed prediction algorithms.

This chapter is structured as follows. Section 4.1 describes in-depth the structure of the study, including a description of the used sequences, the interface used to question the users, and the timing of sequences. Section 4.2 explains how participants were selected for this study and provides demographic information. Finally, section 4.3 presents findings and numbers derived from users' responses.

Conclusions derived from this chapter are exposed in section 4.4.

4.1 Methodology

This study has the goal to ask the question: "Are humans able to predict lane changes?". To do so, social research has been conducted using the PREVENTION dataset as a basis for lane change predictions. This section provides detailed information about the procedure followed in conducting this study. The following subsections describe how sequences were selected, how sequences are presented to users, and how users' predictions are collected.

4.1.1 Sequences

This study has been conducted using sequences extracted from the PREVENTION dataset. According to table 3.3 912 lane-change manoeuvres are recorded in the PREVENTION dataset. However, some lane-change maneuvers were kept out of this study for different reasons. The most common excluded maneuvers were consecutive lane changes. The end of the first lane change could motivate the user's reaction at the beginning of the second one. Despite this, a total of 794 lane changes maneuvers were evaluated in this study. Lane-keeping maneuvers are also included in the evaluation set. A total of 179 lane-keeping maneuvers were manually selected. These maneuvers were selected offering similar situations as those represented in the lane-change set. These situations are commonly overtaking and car-following with or without a posterior lane change. The combination of scenes with different endings produces an unbiased analysis because both actions can be expected after any situation. Otherwise, the user is only concerned about the direction of the lane-change maneuver. Table 4.1 shows the share of each type of maneuver in this study. Note that 83% of the lane changes are developed by using the blinker. This fact reveals that blinker is a heavy conditioner to state a lane-change maneuver.

Table 4.1: Maneuvers count

	NLC	**LLC**		**RLC**	
		w/o blinker	*w/ blinker*	*w/o blinker*	*w/ blinker*
Samples	179	34	203	71	307
		237		378	
%	22.54%	4.28%	25.57%	8.94%	38.66%
		29.85%		47.61%	

The main concern about the number of sequences to be displayed was that the users do not lose interest on the test. Too many sequences can bore the user and provide invalid results. In contrast, few sequences provided less information, and more subjects are needed to achieve the same conclusions. Each user test is composed of a total of 30 sequences. This number was selected using a small number of people, and they found the test not too large and not too short. On average, each sequence takes 15 seconds, depending on the user's reaction time. The complete user test takes no longer than 10 minutes. The set of sequences is composed of an equal number of each maneuver; this is 10 NLC, 10 LLC, and 10 RLC. Sequences are extracted randomly from each subset and displayed in random order. This information is omitted to the study's subjects. Figure 4.3 shows the three types of sequences that are visualized in the test. Typically, NLC sequences show a vehicle from short to far distances where it becomes irrelevant for the ego vehicle. Other samples present vehicles performing a car following action with the choice to overtake. Left lane-change maneuvers are typically caused by overtaking of surrounding vehicles. However, the right lane changes are commonly observed after overtaking the ego vehicle.

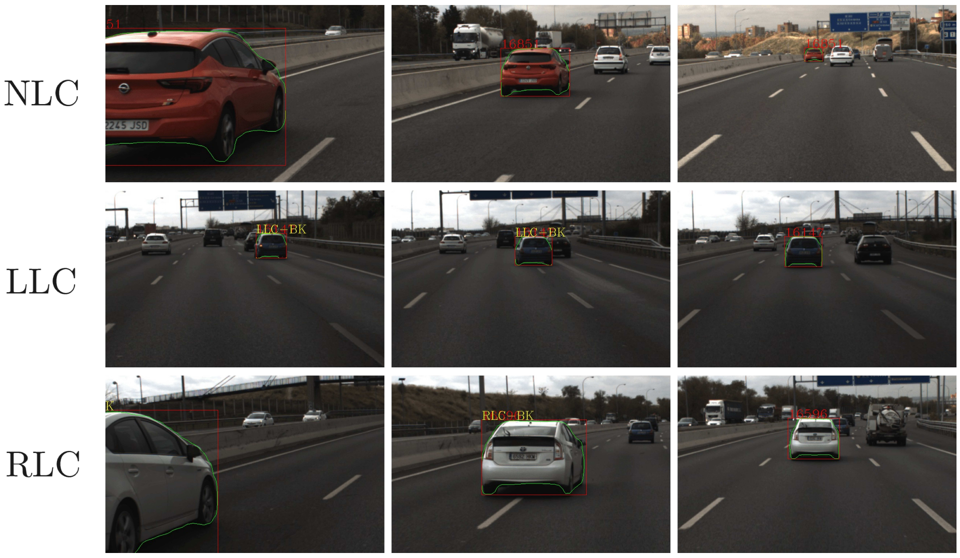
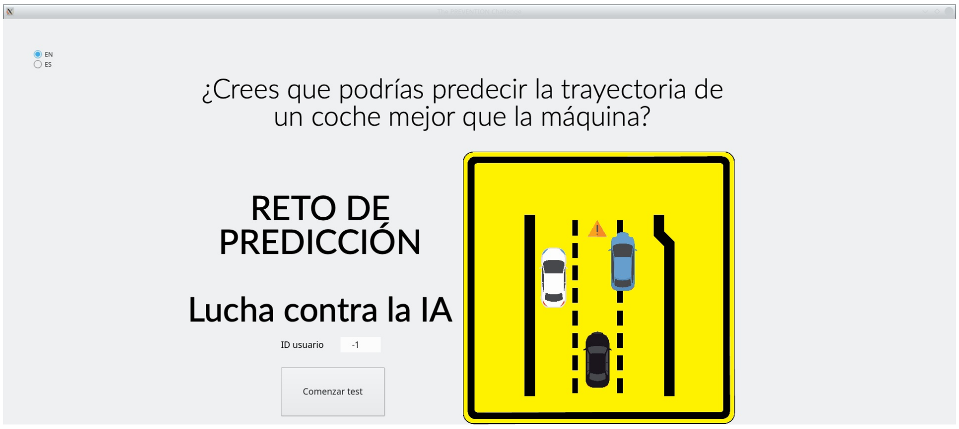


Figure 4.1: Typical sequence representation. No lane-change maneuvers (top) are displayed from beginning to end. lane-change maneuvers (middle and bottom) are represented at the beginning of the maneuver, at the lane change event, and the end of the maneuver.

4.1.2 Interface

The test interface has been developed using a QT application. This interface allows us to generate a random set of sequences, display them, and record the user's responses. This application is composed of the main window, where all the information is displayed, and a pop-up window informs the user for expected inputs from their side. Figure 4.2a shows the Challenge main window. A simple ID box and a *Start* button are available for user interaction. Figure 4.2b and 4.2c shows the two application messages to inform the user. Finally, figure 4.2d shows a frame from an example sequence where the prediction target is marked with a red rectangle. This experiment was conducted in Spain, alternative messages providing the same information were displayed in Spanish if it is desired by selecting the *ES* option in the main window.


[✓ EN] ES

¿Crees que podrías predecir la trayectoria de un coche mejor que la máquina?

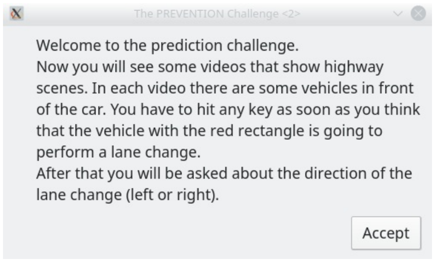
RETO DE PREDICCIÓN

Lucha contra la IA

ID usuario -1

Comenzar test

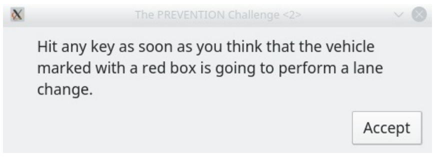
(a) Main window.


The PREVENTION Challenge <2>

Welcome to the prediction challenge.
Now you will see some videos that show highway scenes. In each video there are some vehicles in front of the car. You have to hit any key as soon as you think that the vehicle with the red rectangle is going to perform a lane change.
After that you will be asked about the direction of the lane change (left or right).

Accept

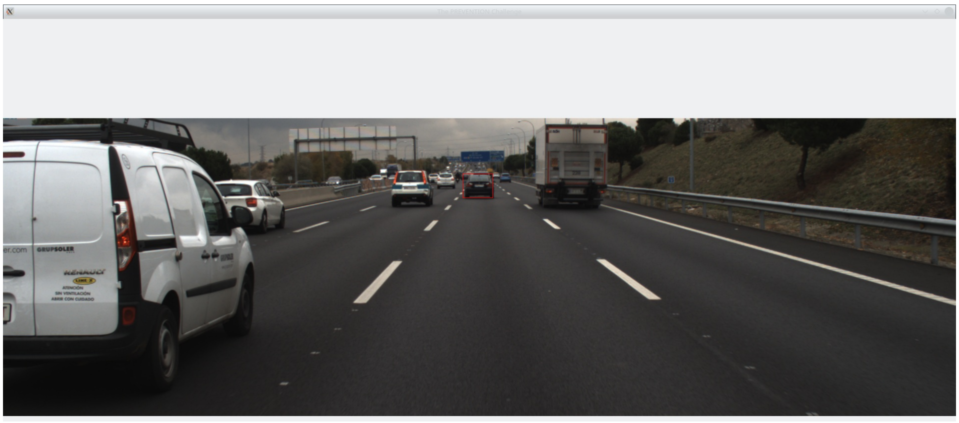
(b) Welcome message.


The PREVENTION Challenge <2>

Hit any key as soon as you think that the vehicle marked with a red box is going to perform a lane change.

Accept

(c) Remember message.



(d) Example of a sequence visualization.

Figure 4.2: Test interface views.

4.1.3 Sequence Visualization

The sequences are video fragments taken from the front camera and displayed at its natural frame rate. The video length has been modified and displayed according to its nature. No lane-change maneuvers are manually selected from beginning to end, and they are entirely displayed adjusted to these limits. Due to user reactions are not expected, no additional actions are required to guarantee variability in this subset. lane-change maneuvers are labeled from the very first frame when the lane change can be observed. A random time is added previously to the lane change to provide some anticipation gap and variability. This period is generated randomly with a uniform distribution function spanning 50 to 100 samples. This random extension guarantees that the user cannot expect lane changes at a fixed time from the beginning of the sequence.

Four values are recorded for each sequence identified as a lane change by the participant:

- seq_0: the frame at which the sequence begins.
- f_0: the frame when the lane-change maneuver has started. This frame is characterized by the triggering of the blinker or the beginning of the lateral displacement.
- f_1: the frame when the lane change event has happened. At this frame, the rear middle part of the vehicle is just between the two lanes. This point is considered the LCE.
- f_u: the frame when the user hits any key. This value represents the frame when the user has detected or predicted the lane change.

Note that no lane-change sequences do not need any temporal reference.

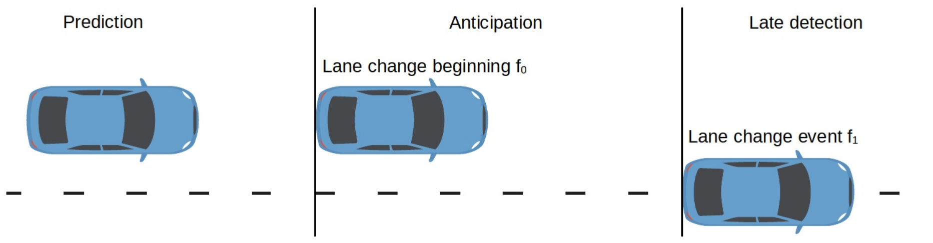


Figure 4.3: Lane change time references.

4.2 Participants

Participants were recruited among the Engineering School, from students to teachers and other research staff, as well as family, friends, and colleagues. Thus, more significant variability is achieved in terms of age, occupation, and participants' driving experience. A total of 72 people did the test in two weeks. They provide some demographic information by filling up a small questionnaire. Then they perform the trial evaluating completely 30 sequences each one. Humans have assessed a total of 22160 sequences, 720 were no lane-changes, and 1440 lane-change maneuvers.

4.2.1 Questionnaire

Subjects were asked to fill a short form before doing the user test. This form has the goal of provided demographic information that could be related to their prediction performance. The form is anonymous. Each user was assigned to an ID, and optionally they can provide their name and email to be contacted with the findings derived from this study. Users were asked with the following form:

- ID, name and email (optional), age and gender (male/female).
- Occupation: study / work / both / none.
- Has driving license: yes - no
  - Driving experience: $\leq$ 1 yr. / 1-2 yr. / $>$ 2 yrs.
  - Driving frequency: daily / weekly / occasionally.
  - Driving areas: urban / highway.

The form is divided into two parts. The first part collects some personal and demographic information. The second part asks the participants about their driving skills and habits. The anticipation of driving behaviors can be closely related to driving skills [65].

4.2.2 User Interaction

In order to avoid biased behaviors from different users, a fixed procedure was established to instruct the volunteers in the test process.

1. The user is asked to fill the form. In the beginning, a unique ID is provided to the user, and it is registered in the questionnaire. After filling it, the user introduces the same ID in the Challenge application.

2. After clicking the *"Start"* button, a message pop-up with a short description of the expected behavior. The message says:

   "Welcome to the prediction challenge. Now you will see some videos that show highway scenes. In each video there are some vehicles in front of the car. You have to hit any key as soon as you think that the vehicle with the red rectangle is going to perform a lane change. After that you will be asked about the direction of the lane change (left or right)".

3. The user is asked to tell the staff what is needed to do. If the answer is satisfactory the user presses the *"Accept"* button. If not, a short explanation with the same information is provided.

4. Before every sequence is displayed, a new pop-up message appears, remembering what to do and waiting for the user to be ready. This mechanism enables a pause system at the user's will. The message says:

   "Hit any key as soon as you think that the vehicle marked with the red rectangle is going to perform a lane change".

5. The user presses the *Accept* button and the sequence is displayed. The vehicle of interest is marked with a red rectangle according to the description provided to the user. The user presses any key if a lane change is predicted or detected. If not, the sequence ends after a few seconds.

   (a) If the user has pressed any key, the sequence finishes immediately, and the screen is cleared. Two buttons appear in the screen to select a *left lane change* or a *right lane change.*

6. If there are remaining sequences to visualize, the process goes back to point 4. After the last sequence, a window pop-up providing some statistics of the user's performance.

   - Accuracy (%). Percentage of correctly labeled maneuvers.
   - Average delay in detecting the lane changes (sec.). Mean delay from the starting of the lane-change maneuver for those correctly labeled.
   - Average lane change anticipation (sec.). Mean anticipation time from the lane change event for those correctly labeled.

7. The main window of the Challenge interface is showed ready for a new user.

4.3 Results

This section reviews the data generated by test participants. Prediction results are provided in terms of accuracy and delays. The existing data is also used to determine if humans can predict lane changes on a regular basis.

4.3.1 Lane Change Prediction Accuracy

How precise humans are predicting lane changes can be evaluated by means of the accuracy, precision, and recall. These values are computed according to eqs. 4.1 and 4.2, where N is the total number of samples, TP are the true positive, FP the false positive, and FN the false negative samples.

$$Accuracy = \frac{TP}{N} \tag{4.1}$$

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \tag{4.2}$$

What is considered a positive or a negative sample depends on different interpretations of users' responses. According to this, four different types of interpretations are carried out attending to the nature of the sequences, if they are correctly labeled or not, and the instant when they were labeled:

- The simplest analysis evaluates correctly and incorrectly labeled sequences. Table 4.2 shows the confusion matrix of all the developed tests. A total of 2160 sequences divided into correct and incorrectly labeled sets. The correct classification maneuvers are a big share of 88.1%. A small number of misclassifications between left or right lanes changes is observed.

Table 4.2: Confusion Matrix I

	Classified Class			
Target Class	***left***	***none***	***right***	**Recall**
left	660	11	49	0.917
none	73	584	63	0.811
right	37	23	660	0.917
Precision	0.857	0.945	0.596	0.881

- The previous analysis reveals excellent human performance. However, it is not the same to state a lane-change maneuver at the very beginning than after the vehicle has crossed the divisor lane. Now, lane changes labeled after the f_1 are considered as no lane-changes. The new confusion matrix is provided in table 4.3. It can be observed that the left lane change detection rate has fallen 5 points, and the right lane change 7.5 points.

Table 4.3: Confusion Matrix II

	Classified Class			
Target Class	*left*	*none*	*right*	**Recall**
left	624	47	49	0.867
none	73	584	63	0.811
right	37	79	604	0.839
Precision	0.850	0.822	0.8436	0.839

- The previous analysis takes as positive samples detections of ongoing lane-change maneuvers. Strictly, predictions are only considered if lane changes are stated before they have started. The same analysis is now conducted considering only the predictions of the lane change as true samples. Table 4.4 shows the metrics for this new consideration. It can be observed that only 15.6% and 10.6% of the left and right lane changes are detected before the lane change begins. This is a dramatic performance reduction concerning the two previous analyses.

Table 4.4: Confusion Matrix III

	Classified Class			
Target Class	*left*	*none*	*right*	**Recall**
left	112	559	49	0.156
none	73	584	63	0.811
right	37	607	76	0.106
Precision	0.504	0.334	0.404	0.327

- The three previous metrics were sequence-based analysis. Now the prediction/classification performance is analyzed in terms of image samples. This special consideration applies only for lane-change maneuvers, where a difference must be done between the different time intervals of the maneuver. Sequences are visualized from a randomly generated sample seq_0 to a fixed frame f_1. The relevant period of analysis is focused from seq_0 to f_0 and from f_0 to f_1. Depending on the user's event, denoted as f_u, samples have different considerations:

- From seq_0 to f_0. This period has the consideration of prediction. Consequently, these samples can be considered as lane change or as no lane-change simultaneously and considered as TP for each corresponding category. From seq_0 to f_u, samples are labeled as no lane-change samples, from f_u to f_0 samples are labeled as lane change (left or right if it is correctly assessed). This is the best case for the user; all the samples are considered correct samples and accounted for each corresponding category. The aim is not to underscore predictions versus detections or vice versa.
- From f_0 to f_1. This period has the consideration of detection. In this period, all the samples must be classified as lane change instances. The samples between f_0 and f_u are labeled as no lane-changes (this happens when $f_u > f_0$). From f_u to f_1, samples are labeled as a lane change in the sense the users suggest. In this phase, the reaction time is punished, indicating wrong classifications.

Table 4.5 shows the classification performance frame-wise. It can be observed that the number of no lane-change is higher than lane changes, 255K versus 54K. The number of correctly lane change classes has reduced significantly, falling to 56% and 53%. This classification metric considers not only the complete sequence classification but also the user's reaction or prediction time.

Table 4.5: Confusion Matrix IV

	Classified Class			
Target Class	***left***	***none***	***right***	**Recall**
left	16450	9595	3028	0.566
none	12153	235799	8627	0.919
right	2475	10051	13997	0.528
Precision	0.529	0.923	0.546	0.853

4.3.2 Lane Change Prediction Delay

Reaction time or prediction time is a fundamental value to assess correctly potential hazardous situations. Even more, lane change predictions after the lane change event are useless. The study's data have been evaluated to provide human anticipation or response time in these specific highway scenarios.

The first parameter we have evaluated is the delay to the beginning of the lane change. This value allows us to determine the level of anticipation or prediction of users. It is considered appropriate to remove the best and the worst values to remove some noise from data. Figure 4.4 shows the average delay of each user to the beginning of the lane-change maneuvers in a sorted way. Negative values represent anticipation or prediction, while positive values represent a delay in detecting maneuvers. The mean users' delay is 1.08 seconds. It can be observed that only four users achieved an average negative delay.

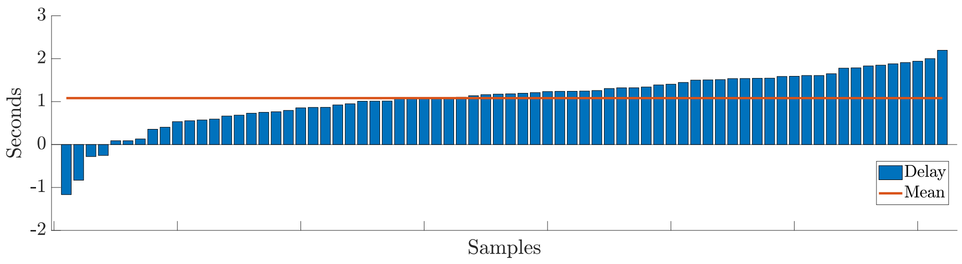


Figure 4.4: User's delay from f_0 event.

The same delay is now evaluated with respect to the lane change event f_1. Note that beyond this point, the maneuver has been completely developed, and its detection has no relevance. Figure 4.5 shows the users' average delay. The mean users' delay is -1.66 seconds. This represents that on average, the lane changes are detected 1.66 seconds before the vehicle crosses the divisor line. All the delays with respect to f_1 are negative due to all sequences detected after f_1 are discarded.

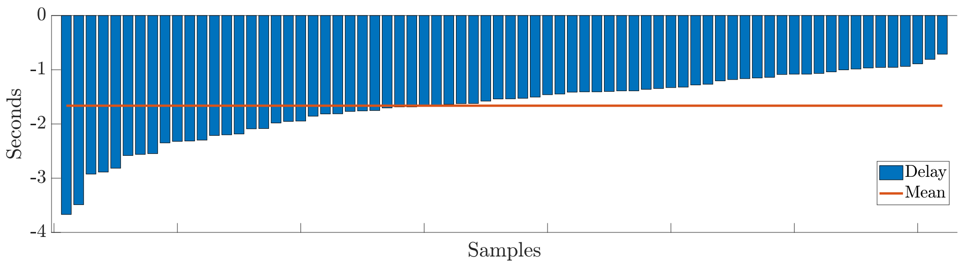


Figure 4.5: User's delay from f_1 event.

4.3.3 Delay vs Accuracy

It is reasonable to think that time-based decisions can become more accurate as much as the decision is delayed. Subsection 4.3.1 shows how lane change accuracy decreases when time limitations are imposed. Subsection 4.3.2 shows a spread range of delays when predicting lane changes. This section provides a correlation between lane change prediction accuracy and delay. Figure 4.6 shows user accuracy versus the prediction delay. Data has been fitted to a 1^{st} order polynomial. The equation parameters reveal that the average accuracy is close to 80% at prediction with a delay equal to zero. Each second earned in prediction loses a 7.6% in accuracy.

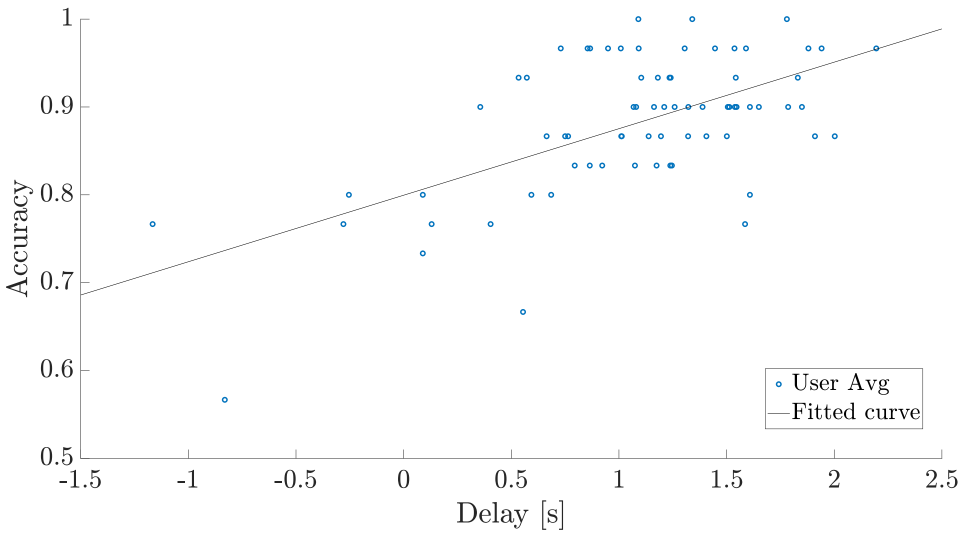


Figure 4.6: Accuracy vs. Prediction delay ($r = 0.076x + 0.799$).

The question "Are humans able to predict lane changes" is now asked using the users' recorded data. The null hypothesis H_0 states that humans are able to predict lane changes. The alternative hypothesis H_a states that humans are not able to predict lane changes. Mathematically, hypothesis H_0 and H_a can be defined as eq 4.3.

$$H_0 : \quad \mu \leq 0 \qquad H_a : \quad \mu > 0 \tag{4.3}$$

The null hypothesis states that the mean value of lane change delay is smaller or equal to 0. The alternative hypothesis establishes that the mean value is higher than zero. The average user delay distribution is characterized by a $\overline{x}$=1.08 and σ_x=0.63 with 72 samples. The hypothesis is analyzed with a confidence value of 99%, or equivalently, a significant value of α=0.01. Due to the small number of subjects, a *T-Student* test has been conducted. The t value that represents our samples is $t = 14.6$. The corresponding *p-value* is 2.225×10^{-23}, almost 0. The null hypothesis can be discarded as far as $p - value < \alpha$. The alternative hypothesis H_a is taken, which means that humans cannot predict lane-change maneuvers systematically.

4.4 Conclusions

This chapter has described the collection of human responses reacting to different traffic maneuvers in the PREVENTION dataset. This data has been used to set a baseline for future comparisons, and to evaluate human performance predicting maneuvers.

Surprisingly, humans cannot predict lane-change maneuvers regularly, at least with the set of sequences used in this experiment. This means that PREVENTION sequences are challenging even for humans. These sequences were recorded in highways during real traffic conditions, which can be expected in the real world. Users' delay predicting lane-change maneuvers has been evaluated. On average, they are detected 1.08 seconds after the lane change has started and 1.66 seconds before the middle point of the vehicle reaches the line between the lanes.

The analysis of the prediction/classification accuracy has been conducted in several different ways. The results vary from excellent 92% in terms of sequences evaluation to questionable 15% in term of sequence predictions. Frame by frame analysis reached 56% of accuracy for lane-change maneuvers.

This study focuses the user's attention on a single target marked with a red rectangle. However, real driving scenarios require to be focused on all the vehicles at the same time, reducing the user's reaction capacity. Real human reaction times can be expected even higher than those observed in this experiment.

Chapter 5

Prediction Models

This chapter describes the developed prediction models. As it was observed in the state of the art, vehicle predictions can be assessed in two different ways. Predictions can be focused on predicting or detecting the type of maneuver, i.e., lane keeping, lane changing, overtaking, cut-in, cut-out, merging ,etc. . . , or focused on predicting vehicle trajectories. Two deep learning predictive models have been developed in this thesis to tackle these two approaches.

This chapter is organized as follows: the maneuver prediction model is presented in section 5.1, found problems, and details of this approach are carefully analyzed here. The trajectory prediction model is precisely described in section 5.2. These proposed methods are deep learning-based approaches. Training details are provided in each corresponding section for a complete description of the whole process. Prediction results, in terms of maneuver or trajectory accuracy, are widely exposed and compared in chapter 6. Finally, conclusions derived from the proposed models are presented in section 5.3.

5.1 Maneuver Prediction

The maneuver prediction or classification problem faces the following situation. Given a scene, a predictive maneuver model must correctly assess all the future maneuvers that have not started yet. Nevertheless, what is this prediction model supposed to do during an ongoing lane change? A classification model must assess ongoing maneuvers correctly. From a practical point of view, the desired behavior of a maneuver-aware system is the combination of both ideas, detecting lane changes as soon as possible while detecting them until the end.

Making use of the PREVENTION dataset, images, vehicle contours, and lane change labels are used to develop the maneuver detection system. This deep learning-based approach is tackled from an image classification perspective. Given a specific input image, a particular output label is desired. The model classifies images into categories. Hereafter, we will refer to the classification term due to the problem's nature, but this is not only limited to the classification of ongoing actions. A future action, not observed at the current moment, can be predicted through classification by labeling the current input image as the future desired action.

5.1.1 Problem Approach

Action recognition and prediction must deal with two problems. One of the problems is the recognition of the prediction target. A single image can present a scene with more than one single vehicle. Simultaneously, some of them can be performing a lane change, and others cannot. So, different outputs must be possible for the same input image. Figure 5.1 shows an image with three vehicles in which one of them is performing a lane change, and the two others are not. In this example, three outputs are desired, specifying which of them is performing the lane change and which are not.

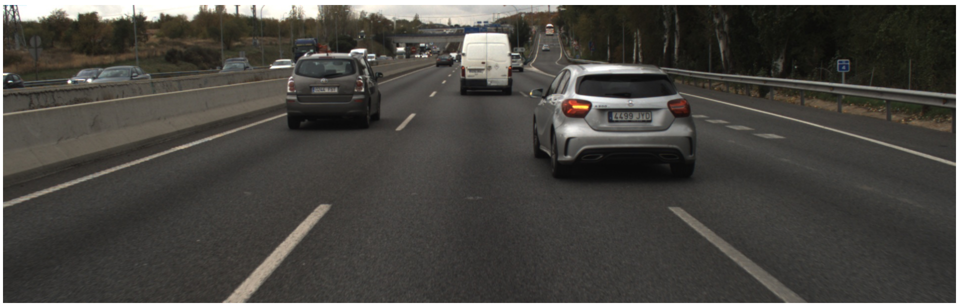
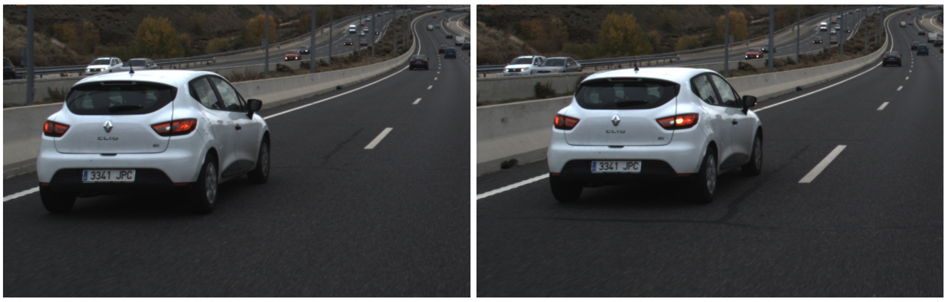


Figure 5.1: Multiple vehicle lane change problem example.

The second problem is the highly temporal dependency of trajectories or lane-change maneuvers. Sequences of images have a better chance to detect time-based actions such as lane changes. Figure 5.2 shows an example of a lane change that apparently cannot be identified as a lane change (figure 5.2a) but it can be in the next frame because of the blinkers (figure 5.2b). An easy solution could be to stack images as a 4D volume, but the problem becomes rapidly computationally infeasible in terms of memory size for training devices.



(a) Image at frame f (b) Image at next frame $f+1$

Figure 5.2: Example of action time dependency.

The solution to the first problem is to use a target selection method. The target selection method draws the shape of the prediction target in a separate image. Figure 5.3 shows the shape of three prediction targets in the scene. Each target selection image is generated using its corresponding contour only. This problem also arose in the user test described in chapter 4 when the user needs to be focused on a specific vehicle. It was worked around selecting the prediction target with a red rectangle in that case.

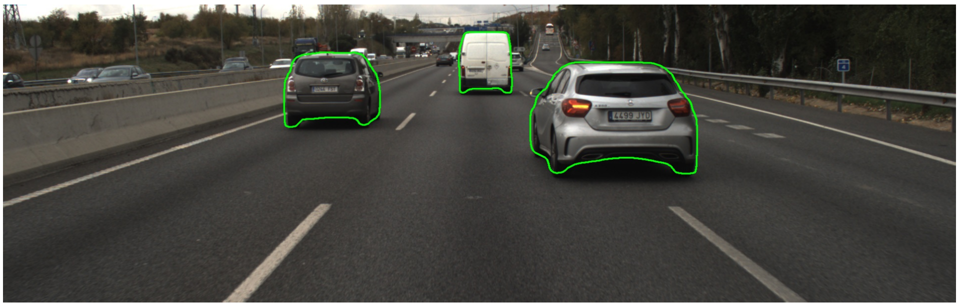


Figure 5.3: Original image with three targets detected.

The second problem can be solved by stacking images across the image depth, but this approach could become computationally infeasible. This problem has been worked around creating a new image with the shape of the vehicles at different time stamps. The time step of each shape is represented using different grayscale values in the image. Figure 5.4 shows an example of this temporal codification.

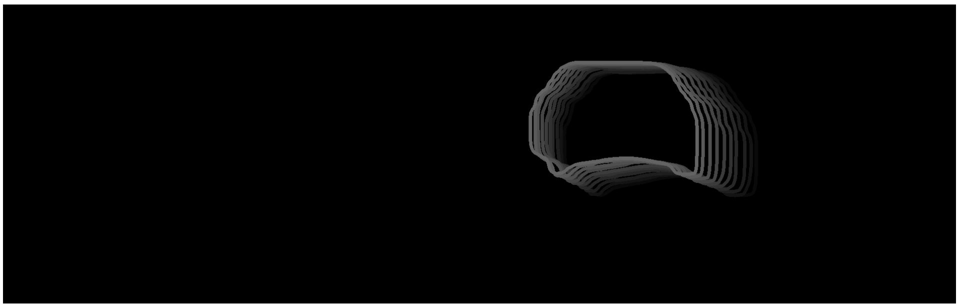
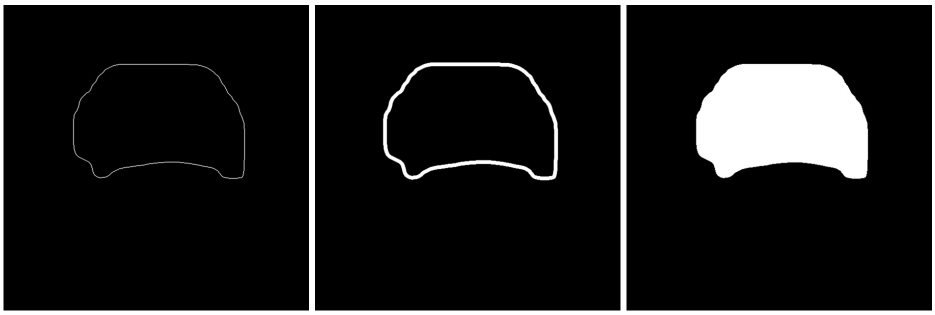


Figure 5.4: Temporal codification of vehicle trajectory.

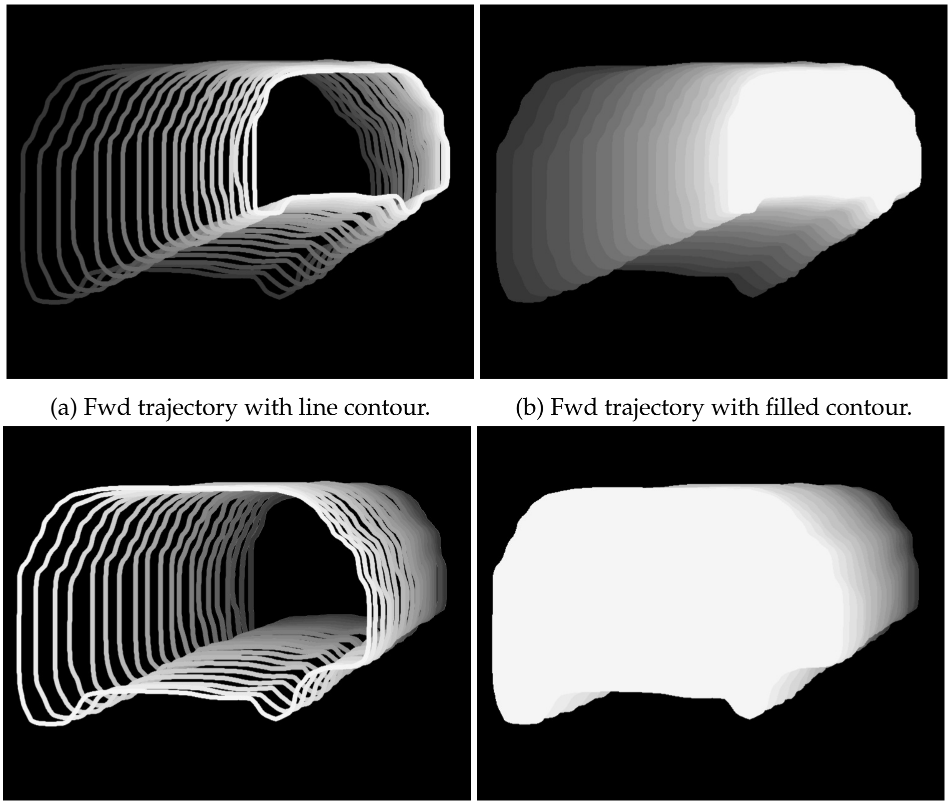
The shape of vehicles can be represented in two different ways: using a contour line or a filled contour. Figure 5.5 shows these types of representation. The width of the line used to draw the contour or if it is filled has an important role in the target selection method and the representation of the vehicle's motion.



(a) Contour line $width = 1$. (b) Contour line $width = 7$. (c) Filled contour.

Figure 5.5: Contour line vs filled contour representation.

The filled contour representation has an important disadvantage with respect to the contour line representation. An oncoming vehicle is seen bigger the closer it is; thus, the contour of the vehicle becomes bigger, and the newer representations partially or totally overlap the older ones. If the position of the vehicle is represented with filled contours, the motion pattern can vanish. However, contour lines minimize the overlap area, and the information persists in the representation. Figure 5.6 illustrates the behavior of the lined and filled contour. The left images (figures 5.6a and 5.6c) show a clear representation with contour lines. In contrast, right images (figures 5.6b and 5.6d) show partially hidden information because of filled contour overlap.


(a) Fwd trajectory with line contour. (b) Fwd trajectory with filled contour.

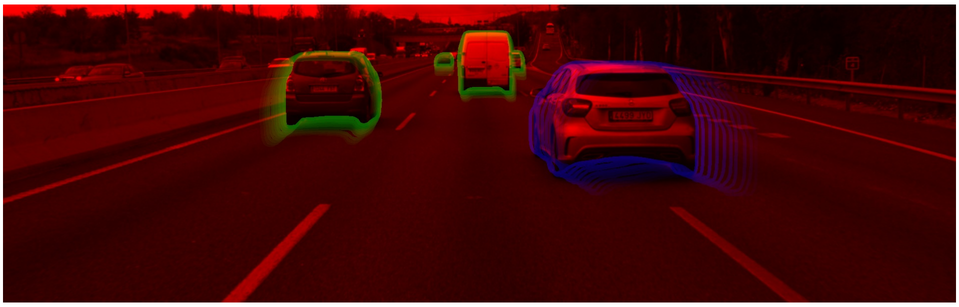
(c) Bwd trajectory with line contour. (d) Bwd trajectory with filled contour.

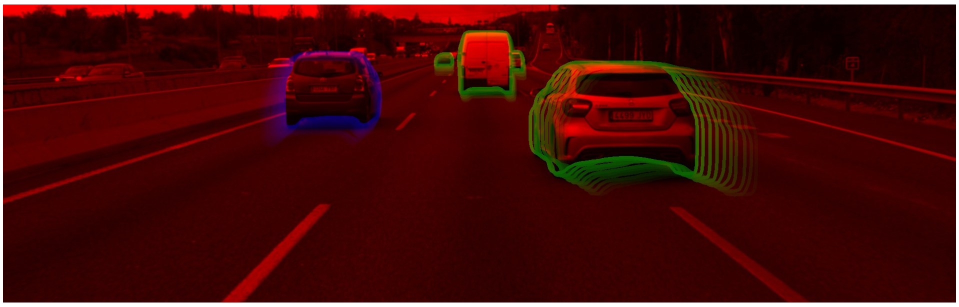
Figure 5.6: Trajectory overlap Contour line vs filled contour representation.

An enriched image is generated for the prediction target combining the target selection method and the motion history representation. This representation can also be used to represent surrounding vehicle motion histories in a separate image. This image, representing surroundings behavior, enables interaction modeling.

Context is present in the three-channel original image, lane markings and road configuration are relevant for a correct scene understanding. However, it can be converted to a gray-scale image, and the information remains while the data size decreases. At this point, three single-channel images that integrate context, target selection, vehicle interaction, and time dependency are generated. They can be combined to compose a new enriched RGB image, as it is shown in Figure 5.7. Each image can now be labeled with the observed action of each prediction target.

(a) Original image.


(b) Vehicle performing a lef lane-change maneuver.


(c) Vehicle performing a lane keeping maneuver.

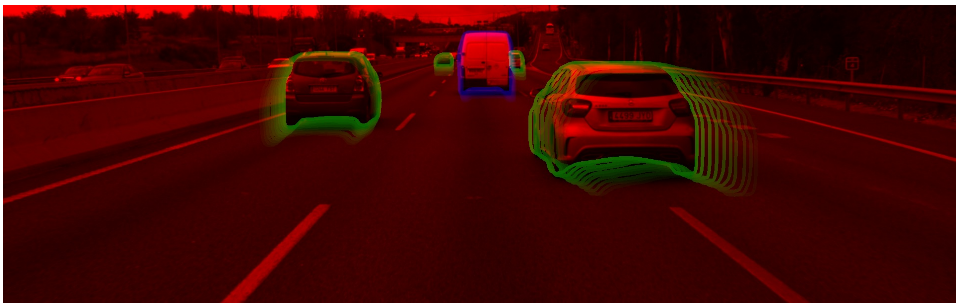

(d) Vehicle performing a lane keeping maneuver.

Figure 5.7: Enriched image composition with target prediction, motion histories, interactions, and context information.

5.1.2 Input and Output Data

According to an image classification approach, the expected inputs are images, and the expected outputs are categories. The procedure described above has been used to generate individual graphic representations for each vehicle at each where it is perceived. The desired output of the classification model is the observed maneuver in the input's image.

The maneuvers considered in this problem are simplified into three categories: *No Lane Change (NLC)*, *Left Lane Change (LLC)*, and *Right Lane Change (RLC)*, where the NLC label is defined by the absence of a lane-change maneuver. The PREVENTION dataset provides labels for each recorded lane-change maneuver. For our interest, lane-change annotations can be described by:

- The ID of the involved vehicle.
- The type of the lane change, that can be left or right.
- The frame when the lane change has started.
- The frame at the LCE, when the middle of the vehicle has reached the divisor line between the lanes.

The ID and the temporal lane change limits allow the labeling of each input image into each corresponding category. As commented before, maneuver classification is directly related to its detection, and maneuver prediction with its anticipation. Making use of the lane change limits the desired anticipation period can also be labeled as lane change samples.

The problem is how to consider positive (lane change) outputs at this anticipation period. Positive samples can be motivated for a precise future lane change prediction or by a wrong estimation of a lane-keeping maneuver. The greater the anticipation period, the higher the uncertainty is. In an extreme situation, if the anticipation period is extended enough, all the situations will end up in a lane-change maneuver because that is what vehicles use to do.

An example of this complex analysis is when a car is overtaking a slower truck on the right lane with no other vehicles in front of it. It is reasonable to think that the car will take the right lane once the truck is overtaken. From a classification point of view, this correct thinking can be considered a success or a fail, only changing the anticipation period.

5.1.3 Training Strategies

This subsection presents the different strategies followed to train the classification model. Two domain strategies are tackled, one based on sample division and others based on sample labeling.

5.1.3.1 Training Set Definition

The scenes presented in the *PREdiction of VEhicle iNTentION (PREVENTION)* dataset are composed of a total of 11 long sequences. These sequences were recorded in highways around the city of Madrid during five different days. Sequences recorded the same day are denoted with the same record number. The different sequences recorded each day are denoted with a different drive number. Sequences recorded the same day has the same hardware setup; this means that the position, orientation, and configuration of the sensors is the same. Table 5.1 summarizes the sequences recorded each day, specifying the area and the type of highway driven and the weather or light condition. The A2 is a stretch of highway between the cities of Madrid and Guadalajara. The M40 and M50 are two rings around the city of Madrid, being the M40 inner to the M50. As ring highways, the road type is mainly straight with a small curvature.

Table 5.1: Sequences and Location Features

Record	Drive	Area	Main Road Type	Light
1	1	A2 - Area 1	Straight	Cloudy
2	1	A2 - Area 2 + M50	Straight + Ring	Dark
	2	A2 - Area 2 + M50	Straight + Ring	Sunny
3	1	A2 - Area 2 + M50	Straight + Ring	Cloudy
	2	A2 - Area 2 + M50	Straight + Ring	Cloudy
4	1	A2 - Area 2 + M40	Straight + Ring	Cloudy
	2	M40	Ring	Cloudy
	3	A2 - Area 2 + M40	Straight + Ring	Cloudy
5	1	A2 - Area 2 + M40	Straight + Ring	Cloudy
	2	M40	Ring	Cloudy
	3	A2 - Area 2 + M40	Straight + Ring	Cloudy

The selection of training and test sets should ideally guarantee the maximum number and variety of examples in both sets. To do so, three strategies are applied to create the training and the test sets:

- Sequence Split strategy: This strategy divides each sequence into two parts, one is used for the training set, and the other is used for the test set. The advantage of this strategy is that all the areas, weather, and sensor setup are included in both sets. The division is done according to a 2/1 training/testing ratio.

- Sequence strategy: The dataset is split preserving the integrity of each sequence. Sequences are entirely used for training or testing. When an area is recorded twice or more, at least one sequence is included in each set. This strategy maximizes the variety of data in terms of areas. Table 5.2 shows which sequences are used to train and which are used to test the models.

Table 5.2: Sequence Strategy Training & Test Subsets

Record	Drive	Training	Test
1	1	✓	
2	1	✓	
	2		✓
3	1		✓
	2	✓	
4	1	✓	
	2		✓
	3	✓	
5	1		✓
	2	✓	
	3	✓	

- One vs. All strategy: This strategy mimics the k-fold method strategy. All the sequences are used as a training set except for one of them, which is used as a test set. This strategy significantly increases the number of samples in the training set. On the other hand, the test set is significantly reduced, and the test results can vary significantly for each fold. This error variance provides an idea of how hard those lane changes are to be predicted.

5.1.3.2 Labeling Strategy

The labeling strategy proposes two methods to label image input samples. The first method is focused on action recognition. Samples from the beginning of the lane change up to the LCE are labeled as lane change samples. The second method is focused on prediction and recognition. Samples from one second prior to the beginning of the lane change up to the LCE are labeled as lane change samples.

The total number of samples in the dataset, these are NLC, LLC, and RLC, are summarized in table 5.3. It can be observed that the number of NLC samples is high compared with the number of lane change samples. The NLC samples represent the 95.4% of the dataset, the LLC the 1.9%, and the RLC the 2.7%. Only NLC samples that were recorded at the same time as lane-change maneuvers have been used for training due to the high number of NLC samples.

Table 5.3: Samples analysis

Event	number
Left Lane Changes	413
Right Lane Changes	499
Left Lane Change samples	17473
Right Lane Changes samples	25420
Lane Keeping samples	876384
Lane Keeping samples while lane changes	148523
Left Lane Change samples + 1 sec. pred	21603
Right Lane Changes samples + 1 sec. pred	30410
Lane Keeping samples while lane changes + 1 sec. pred	177584

5.1.4 Pretrained Models

Predefined state of the art CNNs classification models have been trained to predict and classify images representing highway maneuvers. Table 5.4 shows the set of tested models, from the most complex to the simplest one. A wide variety of models in terms of depth has been tested. The objective is to find the best trade-off between network complexity and the number of samples.

These CNN models are commonly used as the core of different visual problem applications, but these have been designed and trained on image classification problems. They are specifically designed to extract

Table 5.4: CNN models used for Action Recognition and Prediction

Name	Conv Layers	top1-accu	top5-accu
Resnet101	101	78.5%	93.9%
Resnet50	50	78.4%	94.10%
ShuffleNet	50	70.9%	89.8%
GoogleNet	22	–	–
Resnet18	18	59.4%	81.3%
SqueezeNet	18	57.5%	80.3%
AlexNet	8	63.3%	84.6%

high but also low-level information from input images to finally classify them. The final layers of the networks are commonly replaced to adapt these models to specific classification problems. For our problem, all the networks have been appropriately adapted to match a three-class classification problem.

The use of predefined and pre-trained classification models demands an input image with the same dimensions as the original one. The images presented in the dataset have 1920x600 pixels with three channels depth. These images are rescaled to match the size constraints of the used networks, which is 224x224.

The loss function used in the training process is the weighted-cross-entropy. This function allows to weight classes independently and deals with the unbalance problem. The weight of each class is proportional to the inverse of the number of samples. These three class-weight values are normalized to accumulate a total value of one. This mechanism modifies the training process taking more into account minority classes and less the common classes.

5.1.5 Hyperparameters Setup

The training hyperparameters used to train the CNN models are presented in table 5.5. Training parameters have been established experimentally with a special focus on the learning rate parameter.

The learning rate parameter has an important role to play in the training procedure. A high learning rate parameter produces a fast convergence but also a more unstable training process. On the other hand, a small learning rate guarantees a stable training process while increasing the number of iterations and consequently, the training time. The selection of the proper learning rate is critical for successful train-

Table 5.5: Training Hyperparameters

Parameter	**Value**
Optimizer Method	adam
MiniBatch	64
Epochs	2
Shuffle	Epoch
LearnRateSchedule	Epoch Decay
InitialLearnRate	1e-4
LearnRateDropPeriod	1
LearnRateDropFactor	0.1
L2Regularization	0.0001
Momentum	0.9
Epsilon	10e-8

ing. The learning rate has been empirically set accordingly to an experimental analysis of the loss reduction after one epoch training. All the experiments were conducted using the same set of input data and the same initialization of random parameters to achieve valid conclusions. The experiment used 64000 input elements trained in one epoch with 1000 iterations with a batch size of 64. Figure 5.8 shows the cross-entropy loss before and after the training using a different learning rate parameter. The blue line represents the loss before the training, and the red line represents the loss after the training. As can be seen, the optimal learning rate for these experiments is 10^{-4}.

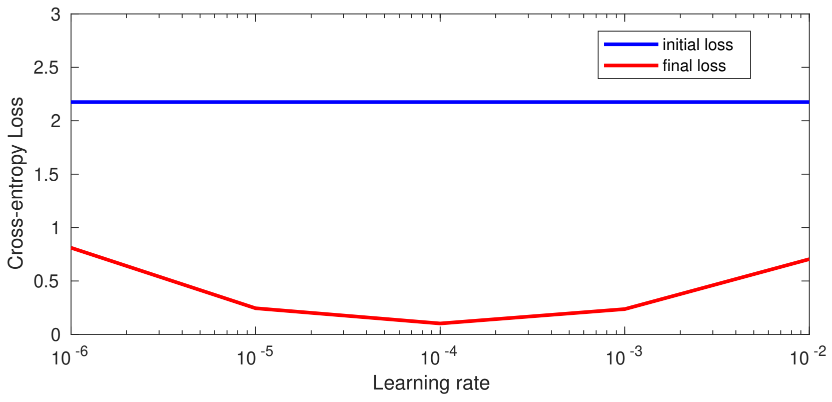


Figure 5.8: Loss reduction vs Learning rate.

5.1.6 Data Augmentation

Additionally, a data augmentation technique has been performed over the input images to avoid overlearning and boost generalization. The technique applies a random translation transformation, by shifting the image vertically and horizontally, to avoid the same input and repeat the same training instance. Image rotation and other transformations have been discarded because they produce non-realistic input images. The random translation parameters are 50 and 30 pixels in the lateral and vertical axes, respectively.

5.1.7 Temporal Output Integration

The CNN generates class-probabilities based on an input image without a temporal relation. However, outputs represent classifications of continuous maneuvers that evolves along the time. The temporal integration of isolated outputs can bring time consistency to the CNNs output. A Markov Chain has been used to integrate consecutive CNN outputs.

The state of the vehicle X is represented by a model with three possible states $X = \{s_0, s_1, s_2\}$, where s_0 is the neutral state which corresponds with a NLC maneuver, s_1 is the LLC, and s_2 is the RLC state. The model is represented in figure 5.9. This model allows transitions from the NLC state to LLC and RLC states and vice versa. However, transitions between LLC or RLC are not allowed. A transition to a middle NLC state is mandatory before changing the sense of the lane change.

The probability of transition between states is defined by matrix T. Each element $T_{i,j}$ represents the probability of change from a state s_i to a state s_j in a discrete time step. Transitions from s_1 to s_2 and vice versa are forbidden or simply not possible according to the model. Consequently, these probabilities must be set as zero $T_{1,2} = T_{2,1} = 0$.

The probability of a certain state s_j at the current time step coming from a state s_i at the past time step is computed based on the probability of the state s_i and the probability of transition between these states according to eq. 5.3, where X_{n-1} is the state in the previous time step. The probability of transit from any state s_j to a certain state s_i is computed as the sum of the probability of being transitioning from a particular s_j to the current s_i according to eq. 5.2.

$$P(X_n = s_j | X_{n-1} = s_i) = T_{i,j} P(X_{n-1} = s_i) \tag{5.1}$$

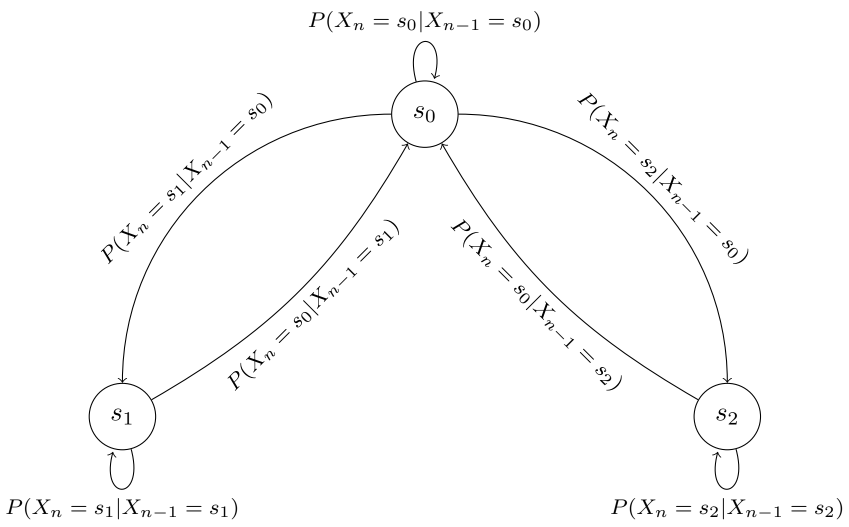

$P(X_n = s_0|X_{n-1} = s_0)$
s_0
$P(X_n = s_1|X_{n-1} = s_0)$
$P(X_n = s_2|X_{n-1} = s_0)$
$P(X_n = s_0|X_{n-1} = s_1)$
$P(X_n = s_0|X_{n-1} = s_2)$
s_1
s_2
$P(X_n = s_1|X_{n-1} = s_1)$
$P(X_n = s_2|X_{n-1} = s_2)$

Figure 5.9: Vehicle behavior model.

$$P(X_n = s_i) = \sum_j T_{i,j} \cdot P(X_{n-1} = s_j) \tag{5.2}$$

In general, the probability of all states can be computed in a matrix way according to eq. 5.3.

$$P(X_n) = T^T P(X_{n-1}) \tag{5.3}$$

Matrix T has been defined based on the frequency of each event observed in the dataset. Note that transitions from a lane change status are observed only twice in a complete lane-change maneuver. Table 5.6 shows the frequency of these events. The probabilities of transition from state s_0, s_1, or s_2 are computed according to eqs. 5.4, 5.5, and 5.6.

Table 5.6: Frequency of Lane Change Events

Event	**Acronym**	number
Number of Left Lane Changes	NLLC	413
Number of Right Lane Changes	NRLC	430
Frames while Left Lane Changes	FLLC	17473
Frames while Right Lane Changes	FRLC	25420
Frames while no Lane Change	FNLC	876384

$$\begin{aligned}
T_{0,0} &= P(X_n = s_0|X_{n-1} = s_0) = \frac{FNLC - NLLC - NRLC}{FNLC} \\
T_{0,1} &= P(X_n = s_1|X_{n-1} = s_0) = \frac{NLLC}{FNLC} \\
T_{0,2} &= P(X_n = s_2|X_{n-1} = s_0) = \frac{NRLC}{FNLC}
\end{aligned} \tag{5.4}$$

$$\begin{aligned}
T_{1,0} &= P(X_n = s_0|X_{n-1} = s_1) = \frac{NLLC}{FLLC} \\
T_{1,1} &= P(X_n = s_1|X_{n-1} = s_1) = \frac{FLLC - NLLC}{FLLC} \\
T_{1,2} &= P(X_n = s_2|X_{n-1} = s_1) = 0
\end{aligned} \tag{5.5}$$

$$\begin{aligned}
T_{2,0} &= P(X_n = s_0|X_{n-1} = s_2) = \frac{NRLC}{FRLC} \\
T_{2,1} &= P(X_n = s_1|X_{n-1} = s_2) = 0 \\
T_{2,2} &= P(X_n = s_2|X_{n-1} = s_2) = \frac{FRLC - NRLC}{FRLC}
\end{aligned} \tag{5.6}$$

It is important to note that T diagonal values, which represent the probability of remaining in the same state, are two orders bigger than the probability of transitioning to another state.

The initial probability $P(X_0)$ is defined using the relative number of occurrences for each state. Eq. 5.7 shows how the initial probabilities vector is computed.

$$\begin{aligned}
P(X_0 = s_0) &= \frac{FNLC}{FNLC + FLLC + FRLC} \\
P(X_0 = s_1) &= \frac{FLLC}{FNLC + FLLC + FRLC} \\
P(X_0 = s_2) &= \frac{FRLC}{FNLC + FLLC + FRLC}
\end{aligned} \tag{5.7}$$

The Markov Chain is used to compute a *priori* probability of the current vehicle state based on its previous state. The joint probability is computed based on the observation provided by the CNN.

5.2 Trajectory Prediction

The prediction of vehicle trajectories deals with the problem of knowing what the vehicle's position some time into the future will be. Vehicle positions are commonly expressed in 2D or 3D cartesian systems. In our case, predictions are tackled from onboard sensors, and vehicle positions are over the road plane. Consequently, a 2D reference system with the origin in the ego-vehicle is considered.

The model proposed to predict vehicle trajectories is detailed in this section. This approach is based on the analysis of graphic representations of the road scene through deep learning techniques. Conceptually, vehicle positions are represented in a BEV image. The same BEV representation is inferred some time ahead by using an image-to-image regression approach.

5.2.1 System Description

The U-net [66] model has been selected as the prediction core to perform the image to image regression task. This model is a kind of CNN, which was developed to perform semantic segmentation in biomedical imagery. U-Net architecture is based on four different blocks: pre-processing, encoder, decoder, and post-processing. Finally, application layers adapt the network's output to the desired topology problem, such as semantic segmentation or image regression.

The U-Net network is defined by a mainstream block that halves the original input image consecutively. Then, the same number of blocks doubles the size of the feature's volume. The features extracted at the input side are concatenated with the features extracted in the output side at the corresponding levels. The number of levels is denoted as depth and represented by the parameter n. Figure 5.10 shows a simplified representation of the U-net's architecture. The pre-processing block generates k features directly from the original image that will be doubled after each depth level.

In this approach, we use this model to perform image to image regression. The scene is represented into a BEV with dimensions $H \times W$. On the input side, d representations of past samples are stacked, creating an image with d channels. At the output side, an image with d channels representing future samples is used as the network target. The idea is that the network core learns the underlying behavior presented in the input block to generate the same representation of the vehicles in some future points.

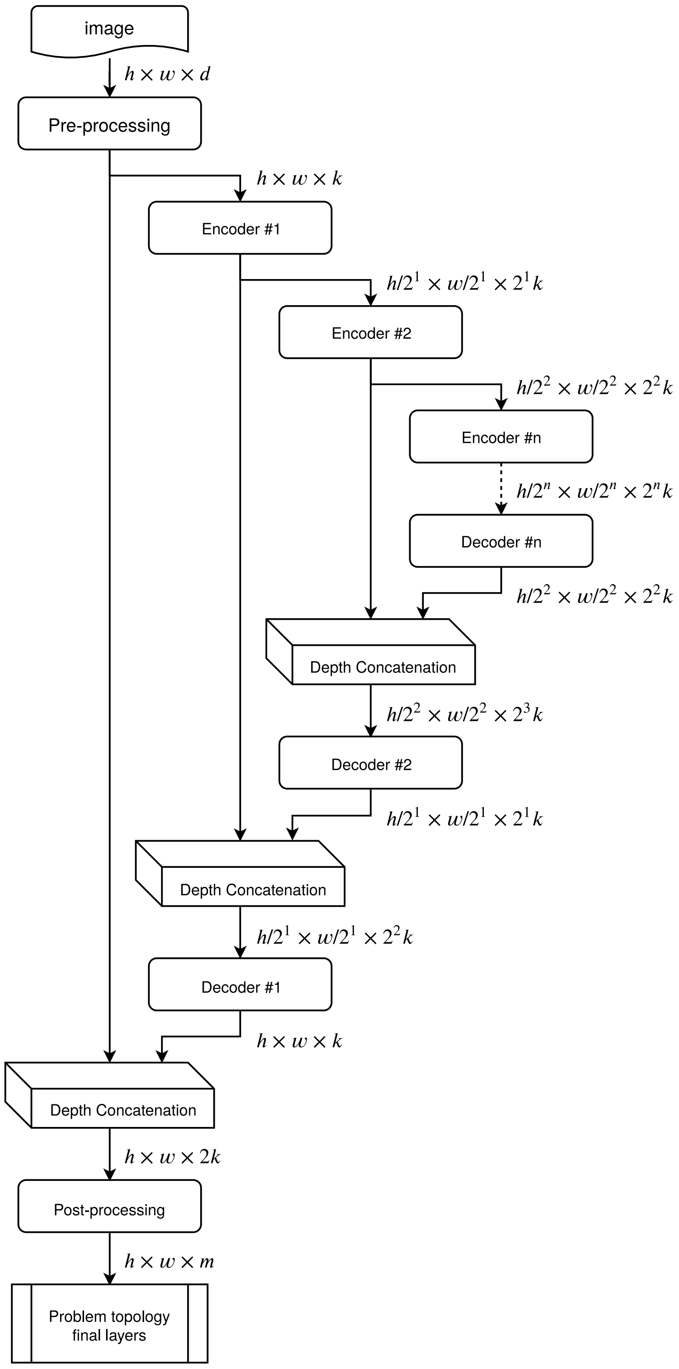


image
$h \times w \times d$
Pre-processing
$h \times w \times k$
Encoder #1
$h/2^1 \times w/2^1 \times 2^1 k$
Encoder #2
$h/2^2 \times w/2^2 \times 2^2 k$
Encoder #n
$h/2^n \times w/2^n \times 2^n k$
Decoder #n
$h/2^2 \times w/2^2 \times 2^2 k$
Depth Concatenation
$h/2^2 \times w/2^2 \times 2^3 k$
Decoder #2
$h/2^1 \times w/2^1 \times 2^1 k$
Depth Concatenation
$h/2^1 \times w/2^1 \times 2^2 k$
Decoder #1
$h \times w \times k$
Depth Concatenation
$h \times w \times 2k$
Post-processing
$h \times w \times m$
Problem topology final layers

Figure 5.10: U-net architecture.

It is important to note that this network takes as input a data block with size at least $2^n \times 2^n$ and multiple of them. Another critical point is to define the receptive field from an output pixel. Eq. 5.8 shows the influence range of a single input pixel where n is the number of encoder-decoder blocks. Each encoder block expands the receptive field with two 3×3 convolution layers, and then virtually multiplies by 2 with a *maxPool* layer with a kernel size 2×2. A decoder block has a *2dtranspose* layer that increases the features by 2 in the two first dimensions. This block doubles the receptive field. Then, two convolution layers with kernel size 3×3 increase the contact surface.

$$r = \pm 2 \left(3 + \sum_{i=2}^{n} 5 \cdot 2^{i-2} \right) \tag{5.8}$$

For example, a U-net network with five depth levels would create a contact surface that links a pixel in the output layer on the position (0,0) with all the pixels in the input layer located at coordinates closer than (156,156). It would connect the output reference pixel to all pixels closer than (316, 316) for six levels of depth. Note that the output reference pixel is in the top left corner. The same connections are created to the left, right, up, and down. This point is critical due to the nature of vehicle interaction, especially in highway scenarios. Driver decisions, and consequently, vehicle trajectories are based on what the driver can see. In highway scenarios, the influence area of a certain vehicle grows according to its speed. The proposed architecture cannot ensure that the inferred vehicle position takes all the driver's visibility into account because of the sensors' range. Table 5.7 summarizes the main features of the U-net model for common depth levels.

Table 5.7: U-net Contact Surface Area

Depth levels	4	5	6	7
Contact Area	±76	±156	±316	±636
Minimum input size	16	32	64	128
Parameters	56k	116k	235k	472k

5.2.2 Input and Output Representation

The input image is generated using the vehicle detections to represent the scene in a BEV representation.

The BEV representation has been defined as a grid with a size of 256×512 pixels representing an area of $\pm 25.6 \times 102.4$ meters in lateral

(y) and longitudinal (x) axis. The size of the grid has been established based on three criteria: memory size when it is allocated in the GPU for training purposes, a proper resolution to understand the scene, and compatibility with the proposed network architecture. The lateral and longitudinal resolutions are 0.1 and 0.2 m/px respectively.

Vehicles can be represented into the BEV using two choices:

- Rectangular representation. It uses a fixed size rectangle $w_v \times h_v$ to populate the BEV whit a fixed value I_v. This representation is closer to reality because of the vehicles' shape.

- Bi-dimensional Gaussian distribution. The Gaussian distribution represents the probability of being a vehicle using a specific tile in the BEV, according to eq. 5.9. The mean value of each bi-dimensional Gaussian distribution μ is set using the vehicle's position. The standard deviation σ is composed with half of the rectangle size.

$$I_v(x,y) = I_v \exp - \left(\left(\frac{x - \mu_{x_i}}{\sqrt{2}\sigma_{x_i}} \right)^2 + \left(\frac{y - \mu_{y_i}}{\sqrt{2}\sigma_{y_i}} \right)^2 \right) \tag{5.9}$$

Rectangle size $w_v \times h_v$ has been set to 1.8×5.0 meters. Equivalently, $\sigma = (0.9, 2.5)$.

At the point where two vehicles' representations overlap, there are two options to merge the area shared by them. They can be added, generating values up to $2I_v$, or they can be limited to the maximum of each vehicle representation according to eq. 5.10. The second option to combine the shared areas represents the real scene in a more reliable way, and the maximum value of the representation keeps limited to I_v.

$$I_v(x,y) = \max_{\forall} \{I_{v,i}(x,y)\} \tag{5.10}$$

Additionally, lanes extracted from the original image can be represented into the vehicle occupancy map to provide context information. The value used to represent the lanes is I_L. If the complete image span (from 0 to 255) is used for representation, in the case of the Gaussian distribution, I_L can match with some points of the vehicle's representation.

Figure 5.11 illustrates a sample of the dataset. Different combinations of Gaussians and rectangle vehicle representations are combined with the lane models provided in the dataset. For this representation,

I_v is set to 128 and 255 for the rectangle and Gaussian representation, and I_L is set to 255.

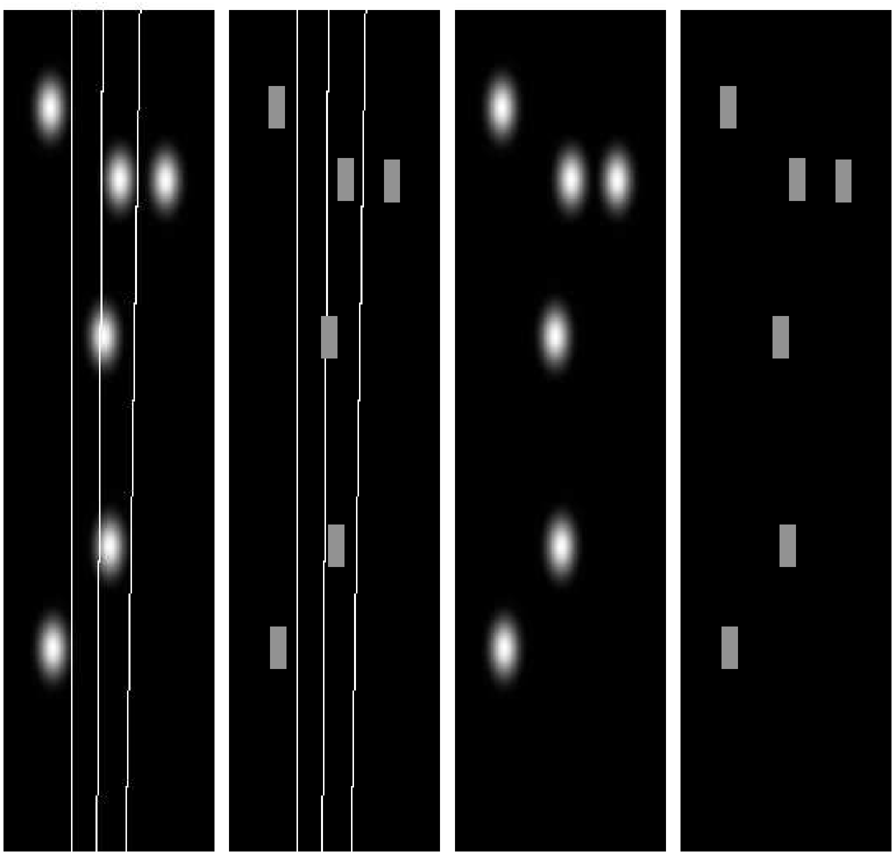

(a) Gauss. w/ lanes (b) Rect. w/ lanes (c) Gauss. w/o lanes (d) Rect. w/o lanes

Figure 5.11: BEV representation of vehicle detections and lanes.

Both input and output images are generated in the same way, but output images do not include lane representations. The complete input and output data consist of d consecutive samples stacked by creating an input/output volume with a size of $256 \times 512 \times d$. When d time samples of data are stacked, a new problem arises in the output block. The output block represents future samples. Consequently, three kinds of vehicles coexist:

- Vehicles that exist in the input and output block. This is the most common case.
- Vehicles that exist in the input block but do not in the output

block. These are vehicles that abandoned the field of view of the sensor.

- Vehicles that do not exist in the input block, but they do in the output block. These are vehicles that enter the field of view of the sensor.

Future positions of vehicles that are not present in the input block cannot be computed as far as there are no data to consider their existence, neither their future positions. Future scene representations are generated considering vehicles that were present in the last input representation only.

5.2.3 Vehicle Position Extraction

The codification procedure transforms numeric data into a representation to make predictions. Once predictions are made, the representation needs to be transformed into numeric data. For each input image, n different predictions are generated at different time horizons. The number of vehicles present in a future scene is a *priori* unknown, so the way used to extract the numeric positions must be able to produce a non-fixed amount of them. It can only consider the vehicles in the future should be the same or fewer than in the last known sample.

The algorithm proposed in 5.12 is used to extract the position of the vehicles. The output data somehow represents the probability of being using a tile in a certain future sample. The algorithm finds the pixel with the highest probability first. This pixel is denoted as $P = (R, C)$, and it is used as the discrete location of the vehicle. The discretization procedure used to represent both input and output data into a grid needs to be reverted. The proposed algorithm extracts the position with sub-pixel resolution in a second step. The position of the vehicle is refined using a scoring function. Each pixel p_i included in the area defined by a rectangle with dimension $R = 2w \times 2h$ around P contributes weighting its probability by its pixel coordinates according to eq. 5.11. Note that discrete positions are conditioned by the resolution used to define the probability occupancy map.

$$\hat{P}(\hat{R}, \hat{C}) = \sum_{r=R-h}^{r=R+h} \sum_{c=C-w}^{c=C+w} p(r, c) \cdot (r, c) \tag{5.11}$$

After computing the sub-pixel position, the area used to compute it is cleared, setting the probability in the occupancy map to zero. This procedure is repeated as many times as tiles with a probability higher than p_{min} remain in the occupancy map. According to the definition of the occupancy map, where a value of 1 means that the pixel is occupied and 0 means empty, we set the threshold p_{min} to 0.5. That is the limit to consider that a pixel represents a possible vehicle. Figure 5.13 shows the codification of an arbitrary vehicle, the red cross represents the actual center of the vehicle, the blue plus symbol represents the discrete found position of the vehicle, and the green one represents the sub-pixel position of the vehicle. Note that the image has been zoomed by 16 to illustrate the differences between discrete and sub-pixel detection. Vehicle parameters are $w = 5.0$, $h = 2.0$, $x = 6.63$, $y = 3.21$, and representation parameters: $x_{ppm} = y_{ppm} = 1$.

Algorithm 1 Multi-Vehicle Location Extraction

```
1: function $\mathcal{P}$ = EXTRACTION($p(r,c), p_{min}, w, h$)
2:     $\mathcal{P} = \emptyset$
3:     while $\exists(r,c) \mid p(r,c) > p_{min}$ do
4:         $P = (R,C)|p(R,C) > p(r,c)\forall(r,c)$
5:         $\hat{P}(\hat{R},\hat{C}) = SubpixelLocation(p(r,c), w, h, P)$
6:         $\hat{P} \in \mathcal{P}$
7:         $\forall(r,c) \in P \pm (h,w)$ do $p(r,c) = 0$
8:     end while
9: end function
10: function $\hat{P}$ = SUBPIXELLOCATION($p(r,c), w, h, P$)
11:     $\hat{R} = \hat{C} = 0$
12:     for $R - h < r < R + h$ do
13:         for $C - w < c < C + w$ do
14:             $\hat{R} + = p(r,c) \cdot r$
15:             $\hat{C} + = p(r,c) \cdot c$
16:         end for
17:     end for
18:     $\hat{P} = (\hat{R}/2h, \hat{C}/2w)$
19: end function
```

Figure 5.12: Position extraction algorithm.

Table 5.8 shows the position extracted from the vehicle shown in figure 5.13. The vehicle is represented in a BEV using the same resolution in both axes, equal to 1 meter per pixel. The position of the vehicle is at coordinates $x, y = (6.63, 3.21)$. The resolution used to make the representation defines the error generated when the maximum method is applied to extract the vehicle's position. However, the subpixel resolution method has errors in the order of tens of millimeters, which is not related to the resolution used to represent the data.

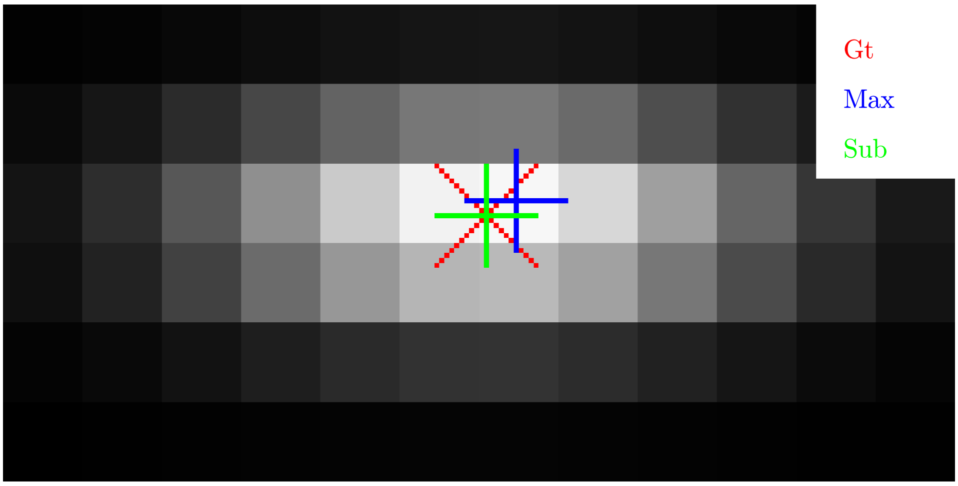


Figure 5.13: Position extraction from graphic representation. The red cross represents the actual center of the vehicle, the blue plus symbol the discrete found position, and the green plus symbol the sub-pixel position. Image augmented 16 times.

Table 5.8: Position Extraction Methods

	Original	Maximum	SubPixel
X / Y [m]/[m]	6.63 / 3.21	7 / 3	6.615 / 3.216
X / Y Error [m]/[m]	- / -	0.37 / 0.21	**0.015 / 0.006**

5.2.4 Association of Extracted Positions

When the positions of all the vehicles are extracted from an image, they need to be associated with their respective detection. A simple procedure based on a Hungarian matrix [61] is used to associate the extracted positions with the positions given in the dataset. The number of elements that can be matched is the minimum between the number of positions extracted from the image of the number of detections provided for the corresponding scene. The value used as the distance parameter to associate elements is the Euclidean distance between extracted points and the provided samples. This method is good enough as predicted positions do not differ from their actual positions.

5.2.5 Training Strategy

The available data in the dataset consists of 11 sequences recorded at 16 Hz. The number of frames accounts for 345K samples representing more than 6 hours of traffic recordings. The amount of data is massive and allows us to train models with a wide variety of samples and situations. The natural frame rate is too high to appreciate relevant differences from one frame to the next one. The original frame rate was reduced from 16Hz to 4Hz, discarding three consecutive samples out of four. The input and output data stack 8 BEV representations of past and future samples, respectively. The lowered frame rate allows the input and output data to cover a larger period using the same number of samples. The time represented in the input block is from t to t - 1.75 seconds. The output block represents future locations from t + 0.25 to t + 2.0 seconds. The network can only be used to predict positions 2 seconds ahead. However, the proposed architecture has the same dimensions in the input and the output sides. The first channel of the output (t + 0.25) can be used as the newest sample (t = 0) in the input block, and the new output extends the prediction one step (t+2.25). This process can be repeated as many times as desired, but the output quality decreases due to the progressive degradation of the input data.

The training set contains samples included in sequences from 1 to 8. Samples from sequences 9 to 11 have been used as a test set.

The influence of high-level parameters such as the network's depth level and the output topology has been tested doing different trainings by varying them. Regarding the depth levels of the U-net, it was limited to 5 and 6. Level 7 and above exceeds the GPU memory size, and the training could not be conducted. Levels below 5 offer a minimal contact area.

The last layer of the U-net was replaced to fit the regression problem. Three different layers were identified as possible output layers: *liner*, *tanh*, and *clippedReLu*. The *linear* layer does not apply a transformation to the network's output. The *tanh* layer applies the hyperbolic tangent function ranging the output into the range of $(-1, 1)$ with a nonlinear transformation. The *clippedRelu* keeps the output between 0 and the given value, which is 1 for this purpose. The last one seems to be perfect to fit the output problem with its values in the range $(0, 1)$. The RMSE has been used as a loss function for the image-to-image regression problem. The main training hyperparameters are listed in table 5.9.

Table 5.9: Training Hyperparameters

Parameter	Value
Optimizer Method	adam
MiniBatch	1
Epochs	1
InitialLearnRate	1e-6
L2Regularization	0.0001
Momentum	0.9
Gradient threshold	1
Epsilon	10e-8

Figures 5.14 and 5.15 illustrate a complete input-output sequence. Vehicles represented as rectangles have been selected for this illustration. Lanes are included at the input block as the way to include context information. The output block has no lane markings given that the desired output is only the vehicles' positions at each corresponding time step.

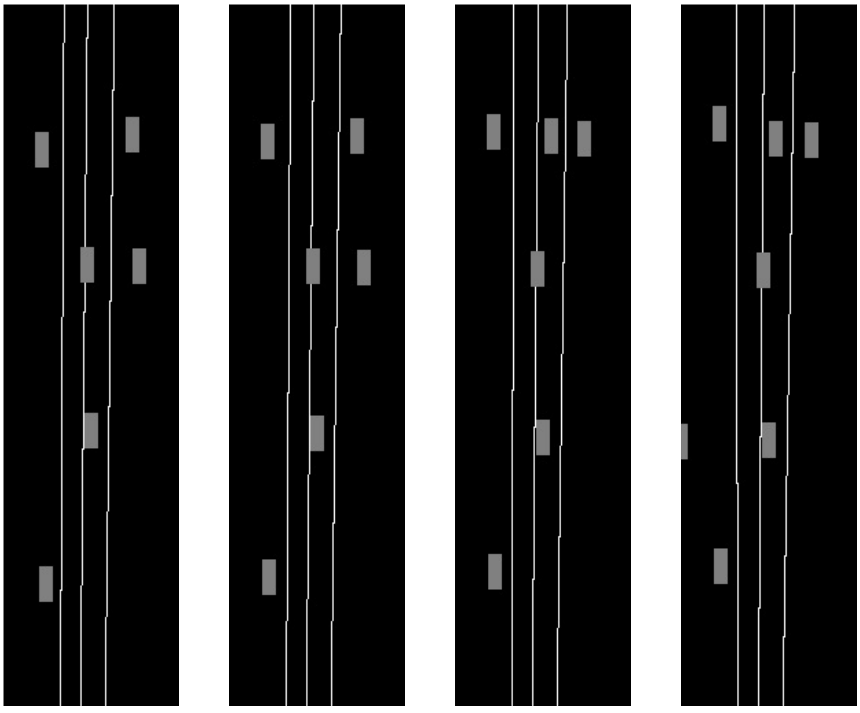
5.2.6 Baseline

For further comparisons, a KF with a *Constant Speed* model has been used to process vehicle positions and generate predictions. State vector X, and the observation vector Y, are defined in eq. 5.12 where x, and y are vehicle positions, v_x and v_y are vehicle speeds, and Δt is the time step. The process model A, and the observation model H are described in eqs. 5.13 and 5.14.

$$X = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^{\mathrm{T}} \qquad Y = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^{\mathrm{T}} \tag{5.12}$$

$$\begin{aligned} X_k &= AX_{k-1} \\ Y_k &= HX_k \end{aligned} \tag{5.13}$$

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.14}$$


(a) $f - 28$ (b) $f - 24$ (c) $f - 20$ (d) $f - 16$

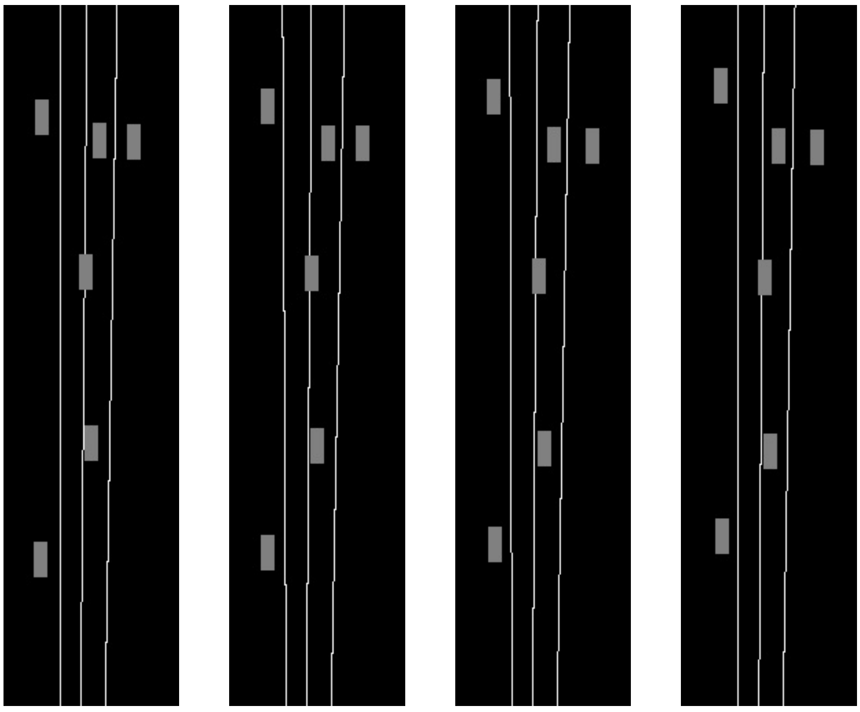
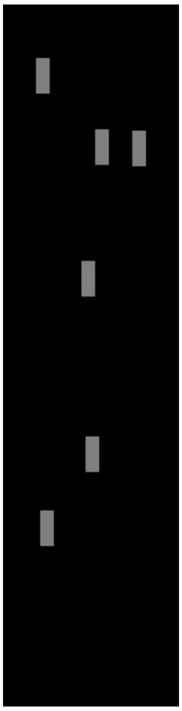
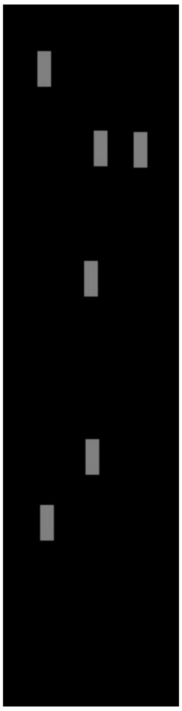
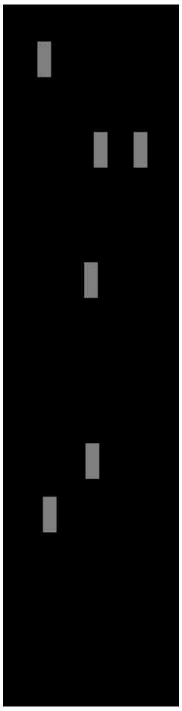
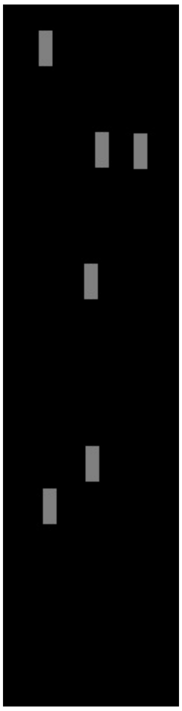

(e) $f - 12$ (f) $f - 8$ (g) $f - 4$ (h) f

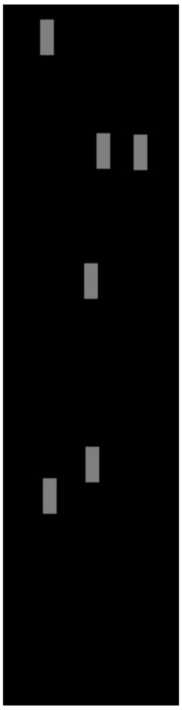
Figure 5.14: BEV input block Sequence. Samples from $f - 28$ to f.

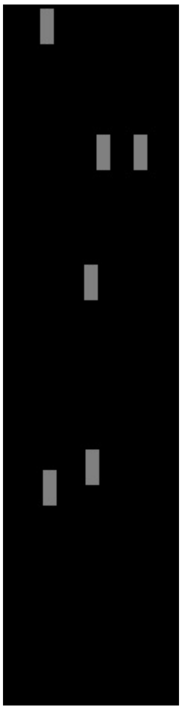

(a) $f + 4$

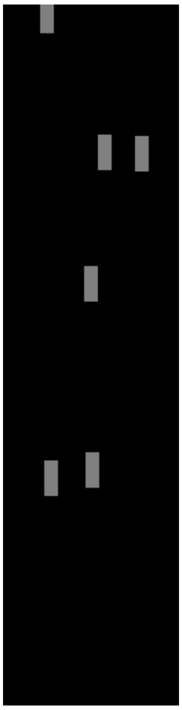

(b) $f + 8$


(c) $f + 12$


(d) $f + 16$


(e) $f + 20$


(f) $f + 24$


(g) $f + 28$

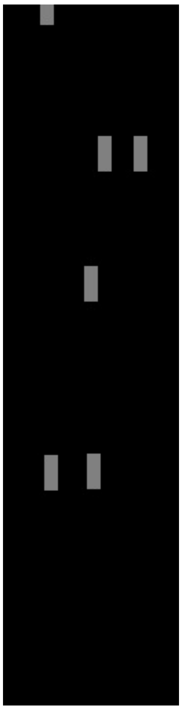

(h) $f + 32$

Figure 5.15: BEV output block sequence. Samples from $f + 4$ to $f + 32$.

5.3 Conclusions

In this chapter, two CNN-based models are proposed to predict maneuvers and trajectories of surrounding vehicles in a deep-learning fashion.

Maneuver predictions are tackled from an image classification approach. Enriched images are efficiently generated to include context information, vehicle interaction, motion histories, and a target selection method. The number of vehicles in the scene and the number of past vehicle representations are virtually unlimited. In contrast with 4D image inputs, the data size does not increase with the number of temporal instances. In contrast with the state of the art approaches, this novel image-based proposal is not limited to a fixed number of vehicles or a fixed vehicle distribution. Besides, none of the existing works uses the appearance of the image to predict or classify the maneuvers of the surrounding vehicles.

A trajectory prediction model using BEV representations has been presented. The model uses vehicle detections and lane information to create a virtual representation of the scene. An image-to-image regression approach is exploited to generate future representations of the current scene. This approach is unlimited in terms of the number of considered vehicles. The trajectory prediction is generated for all the vehicles at the same time, unlike most of the state of the art works, where a single vehicle is considered as the prediction object, and the others actuate as conditions.

Chapter 6

Results

This chapter presents, compares, and discusses the results generated for the models proposed in chapter 5. Section 6.1 presents the maneuver classification and prediction results. The trajectory prediction model is evaluated and analyzed in section 6.2 providing final results. Finally, conclusions derived from this chapter are exposed in section 6.3.

6.1 Maneuver Detection and Prediction

This section presents the results generated by the training and evaluation of the CNN models proposed in chapter 5. The *User Prediction Challenge* described in chapter 4 will be used as baseline for comparison purposes. Following the same structure, results are provided at single-sample and trajectory-wise.

Two sets of data were used to train the models attending to the labeling strategy. The labeling strategy defines the longest prediction horizon by labeling k samples of a future oncoming maneuver. This prediction time is denoted as $t_p = k$. Attending to the value of k two types of strategies are defined. The simplest strategy is limited to the classification of ongoing actions, where $k = 0$. Classification results of current actions are presented in subsection 6.1.1 and denoted as maneuver detection results. The second strategy allows the prediction of an oncoming maneuver up to k samples in the best case. This set is not limited only to the detection of ongoing maneuvers and can also predict oncoming ones. Prediction results of current and future actions are presented in subsection 6.1.2 and denoted as maneuver prediction results.

6.1.1 Maneuver Detection

This subsection presents maneuver detection results. Maneuvers detection is related to the detection of ongoing actions and it is denoted by $t_p = 0$. Classification results can be evaluated in two different ways.

The single-sample classification results evaluate how many samples are correctly categorized. These kinds of results are machine-learning oriented, and they are generated after the models' training. These results are presented in subsection 6.1.1.1.

More complex analysis is performed treating samples as a part of complete maneuvers. Subsection 6.1.1.2 presents maneuver-wise results providing detection rates as well as anticipation or delays.

To find the best configuration, different state-of-the-art CNN models were trained using different sets of data, as it was defined in chapter 5. In this chapter, the different sets are denoted as:

- Sequence Split strategy $S = 1$: half of each record is used for training and the remaining half for testing.
- Sequence strategy $S = 2$: some of the records are completely used for training and the remaining ones for testing.
- One vs. All $S = 3$: follows the K-Fold strategy using each record as the testing set and the remaining ones as the training set.

6.1.1.1 Single-sample Detection

This subsection provides a summary of the maneuver classification evaluating each sample or image as an isolated element. The best way to present the ability of each model to assess the corresponding action for each given sample is the confusion matrix. However, due to the high number of models and variations trained for this purpose, the confusion matrix have been kept apart in the appendix A. Each model configuration has been summarized by its overall accuracy and the by-class precision and recall, according to eqs. 4.1 and 4.2.

Tables from 6.1 to 6.3 show the performance of each CNN model trained with three different sets of data limited to maneuver classification only. It can be observed that Resnet50 and Resnet101 models achieved almost similar scores for the training sets S_1 and S_2. However, when S_3 is used as training set, which implies many more samples, the Resnet50 overcomes all the other models.

Table 6.1: Models Results Summary, $S = 1$, $t_p = 0$

		AlexNet	SqueezeNet	GoogleNet	Resnet18	Resnet50	Resnet101
	Accuracy	72.2	76.3	76.3	72.2	**84.2**	**84.3**
none	**Precision**	75.7	78.1	78.1	75.7	93.8	91.3
	Recall	92.0	92.6	92.6	92.0	87.9	89.8
left	**Precision**	58.7	68.0	68.0	58.7	47.5	56.5
	Recall	32.4	43.7	43.7	32.4	63.0	60.0
right	**Precision**	62.5	72.2	72.2	62.5	57.3	65.0
	Recall	44.3	48.9	48.9	44.3	71.6	68.6

Table 6.2: Models Results Summary, $S = 2$, $t_p = 0$

		AlexNet	SqueezeNet	GoogleNet	Resnet18	Resnet50	Resnet101
	Accuracy	72.3	76.8	75.1	86.3	**86.9**	**87.0**
none	**Precision**	70.9	76.5	75.1	93.0	90.5	93.1
	Recall	95.8	95.2	94.2	91.1	91.1	91.6
left	**Precision**	69.1	74.7	74.7	54.1	53.2	63.1
	Recall	38.8	40.9	37.4	67.1	66.9	61.3
right	**Precision**	83.6	80.3	75.1	68.0	70.0	66.0
	Recall	40.2	49.2	47.6	67.2	71.1	75.9

Table 6.3: Models Results Summary, $S = 3$, $t_p = 0$

		AlexNet	SqueezeNet	GoogleNet	Resnet18	Resnet50	Resnet101
	Accuracy	73.8	76.8	77.8	85.5	**86.9**	82.7
none	**Precision**	74.9	77.2	79.8	93.0	93.4	88.6
	Recall	93.3	94.3	93.0	90.0	91.1	91.7
left	**Precision**	66.8	74.3	67.4	54.3	57.9	57.9
	Recall	36.4	40.7	43.0	62.2	65.3	46.7
right	**Precision**	72.4	76.1	73.2	63.2	68.8	64.1
	Recall	45.9	51.0	51.0	70.7	74.0	62.1

Model Resnet50 trained with set S_3 achieved the best classification results. Table 6.4 presents results for each training fold, for a deeper analysis of the differences between the training sets. Attending to the accuracy as an average value of the performance for each training, folds 4, 8, and 11 look like the three most challenging records (red) due to their low accuracy. On the other hand, folds 7, 9, and 10 seem to be the easiest ones (green). According to the *User Prediction Challenge*, humans asses 85.3% of the frames correctly on average as it is presented in table 4.5. Resnet50 model, trained with the K-Fold configuration achieves 86.9% accuracy on average which overcomes humans' accuracy a 1.6%.

Table 6.4: K-Fold Test Results: Resnet50, $S = 3$, $t_p = 0$

Test Fold		1	2	3	4	5	6	7	8	9	10	11	All
	Acc	86.4	85.8	85.7	**82.9**	86.3	86.2	**88.9**	**84.5**	**88.1**	**87.5**	**84.0**	86.9
none	**Pre**	87.2	90.2	92.6	87.1	89.8	90.9	93.3	90.5	94.0	90.1	87.9	91.1
	Rec	96.1	92.3	90.4	92.1	95.1	92.8	93.9	89.9	91.9	95.5	91.7	93.4
left	**Pre**	72.7	65.1	64.2	74.9	73.5	42.3	70.1	64.6	64.4	72.5	73.3	65.3
	Rec	41.4	62.4	62.7	51.7	59.0	54.9	62.4	51.1	61.5	59.2	58.6	57.9
right	**Pre**	87.1	76.9	66.1	71.3	74.5	79.7	68.3	67.3	70.3	75.9	65.8	74.0
	Rec	70.0	70.0	76.7	74.6	64.4	61.7	72.2	78.8	80.5	57.7	65.7	68.8

6.1.1.2 Maneuver-level Detection

The previous subsection analyses the CNNs' outputs as an isolated element. However, lane change and lane-keeping maneuvers are time-based actions and they should be treated as they are. The raw output of each maneuver sequence is temporarily filtered by using the Markov model described in subsection 5.1.7.

The procedure used to evaluate the maneuver detections is as follows:

- A lane-change maneuver is considered as a correct detection if it is correctly stated before the LCE takes place.
- A lane-change maneuver is considered as a wrong detection if the lane change is not detected or if it is detected after the LCE.
- A lane-keeping maneuver is correctly detected if all of their samples are classified as *none* samples.

To evaluate all the lane-change maneuvers commonly, their length has been normalized to 1. The normalization prevents weighting effects between long and short maneuvers.

Tables from 6.5 to 6.7 presents average numeric results for each trained model. Anticipation is provided in seconds to the LCE and relative to the maneuver's length, where 100% means the maneuver is correctly detected from its beginning and 0% at the LCE. The *Area Under the Curve (AUC)* represents in a single number the ability to detect as soon as possible all the maneuvers, including those which are not detected. Finally, accuracy is included in tables to introduce the binomial anticipation versus accuracy. Accuracy for the LC set, which includes left and right lane changes, and the NLC set is provided. These two accuracy values refer to the original training set. The balanced set refers to the set used in the *User Prediction Challenge* which is the reference to be compared.

Table 6.5: Maneuver Anticipation vs Accuracy $S = 1, t_p = 0$

	Anticipation		**AUC**	**Accuracy**			
	[s]	[%]		LC	NLC	All	Balanced
Alexnet	2.53	81.0	0.67	78.4	68.7	69.9	75.2
GoogleNet	2.45	77.8	0.69	83.7	71.6	73.1	78.3
SqueezeNet	2.54	81.3	0.75	87.8	68.0	70.4	79.3
Resnet18	2.23	68.9	0.54	73.0	86.8	85.1	77.8
Resnet50	2.16	67.2	0.52	72.1	87.9	85.9	77.1
Resnet101	2.28	72.2	0.61	80.3	84.2	83.7	81.8

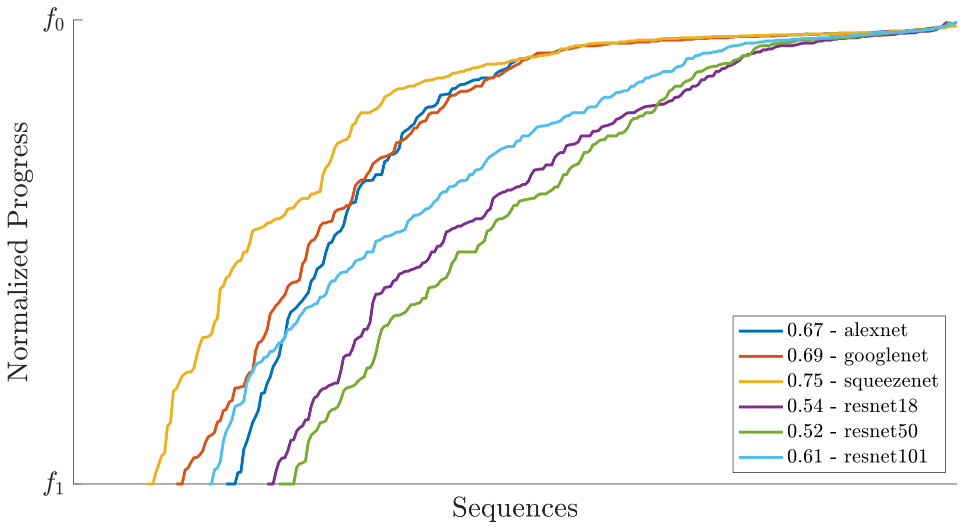


Figure 6.1: Normalized lane change detection representation. $S = 1, t_p = 0$.

Figures from 6.1 to 6.3 represents graphically the performance of the trained models detecting lane-change maneuvers. Each maneuver is sorted based on the detection time from the latest to the earliest. lane-change maneuvers that were not detected before the LCE are placed on the left side of the graph, with an equivalent detection frame to f_1. The early detections are located on the right side of the figure where higher levels of anticipation are achieved. The AUC value is extracted from this representation where non-detected maneuvers account for zero and the detected ones for their corresponding normalized anticipation. Models can be visually compared, better ones described most-left and most-high curves. Note the point where each curve starts defines the share of non-detected maneuvers.

Table 6.6: Maneuver Anticipation vs Accuracy $S = 2$, $t_p = 0$

	Anticipation		**AUC**	**Accuracy**			
	[s]	[%]		LC	NLC	All	Balanced
Alexnet	2.36	82.6	0.77	90.9	65.3	68.0	83.1
GoogleNet	2.36	82.5	0.78	89.7	65.9	68.4	82.3
SqueezeNet	2.37	83.2	0.79	90.9	67.3	69.8	82.3
Resnet18	2.03	70.8	0.61	82.9	86.5	86.1	84.0
Resnet50	2.01	70.2	0.61	80.6	87.2	86.5	82.7
Resnet101	2.09	74.2	0.65	82.9	86.3	86.0	83.9

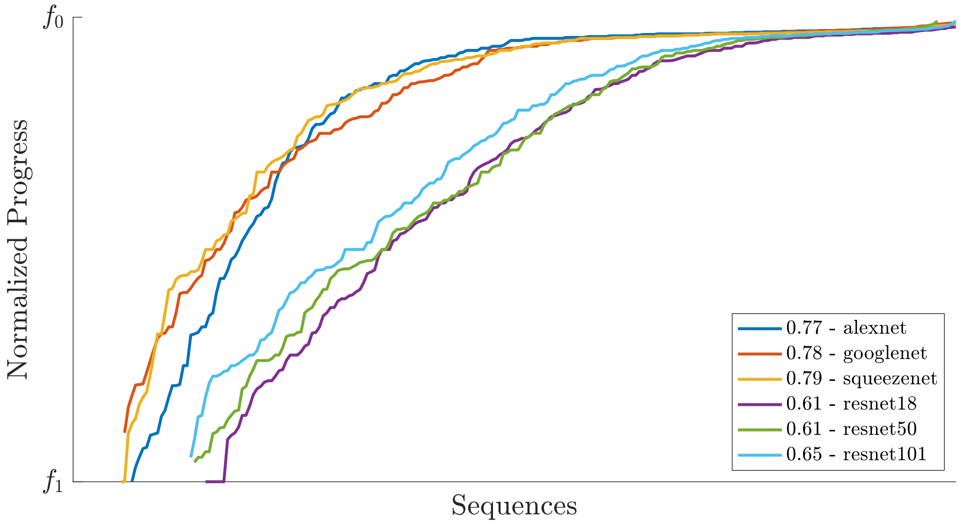


Figure 6.2: Normalized lane change detection representation. $S = 2$, $t_p = 0$.

Comparing the three tables and figures it can be observed that Alexnet, GoogleNet, and SqueezeNet models are better than Resnet models in detecting maneuvers. This fact suggests that the simplest models are better in detecting maneuvers than the most complex ones. SqueezeNet's anticipation goes from 2.37 to 2.53 seconds corresponding to training sets S_2 and S_1. Compared to Resnet50, which achieved the best single-sample classification results, anticipation goes from 2.00 to 2.15 seconds for the same sets.

If the AUC is considered as a performance parameter, the simplest models overcome the most complex again. Between the three simplest models, SqueezeNet achieves 0.79 AUC using training set S_2 and 0.78 with S_3.

Table 6.7: Maneuver Anticipation vs Accuracy $S = 3$, $t_p = 0$

	Anticipation		**AUC**	**Accuracy**			
	[s]	[%]		LC	NLC	All	Balanced
Alexnet	2.33	78.9	0.72	87.9	65.4	68.0	80.9
GoogleNet	2.40	81.0	0.75	88.8	68.4	70.7	81.7
SqueezeNet	2.43	83.3	0.78	90.9	66.6	69.5	82.9
Resnet18	2.16	74.1	0.62	80.0	85.1	84.5	81.6
Resnet50	2.09	72.6	0.66	87.3	84.5	84.8	86.4
Resnet101	2.28	77.3	0.66	79.7	79.0	79.1	80.9

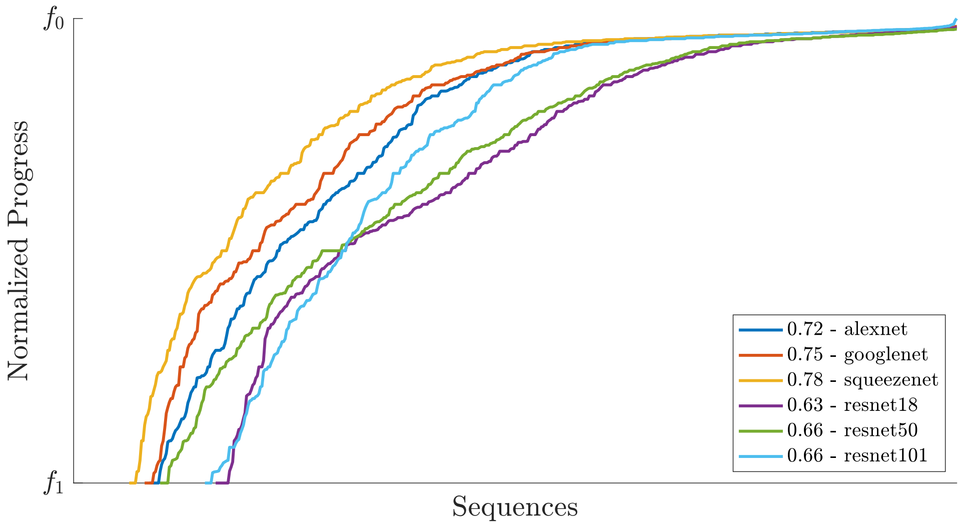


Figure 6.3: Normalized lane change detection representation. $S = 3$, $t_p = 0$.

There is a relevant effect regarding accuracy. While the simplest models achieve higher accuracy for lane-change maneuvers, they are worse in detecting lane-keeping maneuvers (90.9 vs 66.7, SqueezeNet table 6.7). On the other hand, Resnet models achieve quiet similar accuracy for lane change and lane-keeping maneuvers (87.3 vs 84.5, Resnet50 table 6.7). This situation is generated by the imbalance number of maneuvers in the training sets. The additional *Balance* accuracy column shows the accuracy for the *User Prediction Challenge* in which the amount of no lane-change maneuvers is similar to the lane change ones. In this particular set, the Resnet50 model overcomes all the other models.

This behavior together with the anticipation and the single-samples classification results suggests that the simplest models (AlexNet, GoogleNet, and SqueezeNet) are more sensitive and reactive to small variations while Resnet models are less sensitive in general.

Comparing the human and the model's ability to assess maneuvers correctly, humans reached 83.9% accuracy (see table 4.3). Alexnet, GoogleNet, SqueezeNet, Resnet18, and Resnet101 models reached accuracy levels classifying maneuvers (see table 6.7) below the human ability. In contrast, all the models overcome human anticipation (1.66 seconds on average). It is observed that models with higher anticipation periods have lower accuracy in detecting maneuvers. This behavior was observed among the different users that performed the test. Resnet50 model overcomes human's accuracy and anticipation in 2.5% and 0.43 seconds respectively.

6.1.2 Maneuver Prediction

Results presented in the previous section strictly stick to the recognition of ongoing maneuvers. Models were trained and tested using samples corresponding to ongoing lane changes or no lane-changes. However, some of the lane-change maneuvers were detected from their very beginning (just at f_0). To perform predictions through CNN classification architectures, a small number of samples before the lane-change maneuvers were labeled as their corresponding future maneuver. The period used to extend the prediction has been limited to 10 samples (1.0 second) due to high uncertainty related to future action classifications. The behavior expected from the predictive models is to classify lane change actions before they have started but also while they are carried out. This behavior creates a fork in the training process where the no lane-change samples must be classified as they are but on the other hand, no lane-change samples before a lane change must not. In other words, the period before an action transition can be understood as a present maneuver detection or as a future maneuver prediction. The limitation of the prediction period reduces the uncertainty and focuses the models to generate the desired predicted actions. Otherwise, the predictions performed by any model can fail systematically due to the lack of motivated information.

The maximum prediction period is denoted as $t_p = 10$ where 10 samples where added before each lane-change maneuver. The no lane-change maneuvers were also extended 10 samples from the beginning. Following the same structure presented in subsection 6.1.1 results are analyzed as single-sample results, which are machine-learning oriented in 6.1.2.1, and at trajectory level, providing maneuver detection rates, anticipation, and prediction periods in 6.1.2.2. The trained models and the data used for this purpose are the same as those used in the previous subsection except for the lane-change maneuver's length as the only significative difference.

6.1.2.1 Single-sample Prediction

This subsection provides a summary of the maneuver classification evaluating each sample as an isolated element. Each model configuration has been summarized by its overall accuracy and the by-class precision and recall, according to eqs. 4.1 and 4.2. Each specific confusion matrix have been kept apart in appendix A for simplicity.

Tables from 6.8 to 6.10 show the performance of each CNN model trained with three different sets of data including classification of current and future maneuver state samples. It can be observed that there are no significant differences between Resnet18, Resnet50, and Resnet101 models. Compared with AlexNet, SqueezeNet, and GoogleNet their performance is from 10 to 12 points lower. It can also be observed that training set S_2, which implies much fewer samples than S_3, produces slightly better results with the Resnet models, unlike the simpler models.

Model Resnet50 trained with set S_2 achieved the best classification results. The performance between S_2 and S_3 decreases 1.5 points for this model. It was expected to have a better performance over the S_3 set because it includes more samples to train the model. This fact suggests the test samples in set S_2 are easier than test samples in S_3, or the training samples in S_2 are more valuable, learning-wise, than those in S_3 which is really improbable because S_3 set is trained with

Table 6.8: Models Results Summary, $S = 1$, $t_p = 10$

		AlexNet	SqueezeNet	GoogleNet	Resnet18	Resnet50	Resnet101
	Accuracy	71.0	70.8	73.0	**82.9**	**82.6**	**82.8**
none	**Precision**	75.0	70.4	75.3	93.6	93.0	91.3
	Recall	91.3	93.5	91.0	86.5	86.9	88.3
left	**Precision**	48.9	69.0	59.9	44.2	45.0	44.4
	Recall	31.7	36.3	38.3	58.7	57.7	62.5
right	**Precision**	64.6	74.5	69.4	50.7	51.8	62.6
	Recall	40.3	43.1	44.6	70.6	82.6	82.8

Table 6.9: Models Results Summary, $S = 2$, $t_p = 10$

		AlexNet	SqueezeNet	GoogleNet	Resnet18	Resnet50	Resnet101
	Accuracy	72.0	70.8	73.0	**85.1**	**85.9**	**85.6**
none	**Precision**	73.1	70.4	75.3	92.0	93.1	92.6
	Recall	94.3	93.5	91.0	90.6	90.5	90.7
left	**Precision**	63.2	69.0	59.9	53.4	52.5	50.0
	Recall	29.7	36.3	38.3	58.6	63.2	62.7
right	**Precision**	72.0	74.5	69.4	64.6	64.6	67.5
	Recall	45.2	43.1	44.6	66.2	68.7	85.6

Table 6.10: Models Results Summary, $S = 3$, $t_p = 10$

		AlexNet	SqueezeNet	GoogleNet	Resnet18	Resnet50	Resnet101
	Accuracy	73.1	75.6	76.0	**84.2**	**84.4**	**84.1**
none	**Precision**	75.6	76.3	78.1	92.5	92.3	90.1
	Recall	91.8	93.8	92.5	89.0	89.3	90.9
left	**Precision**	58.8	71.8	61.9	49.7	51.4	58.2
	Recall	34.7	39.5	40.7	59.0	59.0	53.5
right	**Precision**	67.8	74.0	73.8	59.6	60.7	65.0
	Recall	43.6	48.4	47.5	67.7	68.2	65.0

all the samples except those included in the testing fold. However, as it was commented before, only S_3 set ensures that all the samples are evaluated and comparable with the results derived from the *User Prediction Challenge*.

Table 6.11 presents results for each individual training fold, for a deeper analysis of the differences between the training sets. Attending to the accuracy as an average value of the performance for each training, folds 1, 8, and 11 look like the three most challenging records (red) due to their low accuracy. On the other hand, folds 7, 9, and 10 seem to be the easier ones (green).

According to the *User Prediction Challenge*, humans' asses 85.3% of the frames correctly on average as it was presented in table 4.5. The Resnet50 model, trained with the K-Fold configuration achieves 84.4% accuracy on average which is 0.9% below humans' accuracy.

This number does not mean the model performs worse than humans assessing current and future maneuver state based on a single sample. Note that all the samples before the lane-change account as correctly categorized samples independently on whether the user predicts or not the future maneuver. The training process does not allow this behaviour and the confusion matrix shows each sample as it is. If the same criterion is applied to the CNN's output the classification results for Resnet50 model trained with S_3 set increases its accuracy up to 87.2% performing 1.9 points over human's accuracy.

Table 6.11: K-Fold Test Results: Resnet50, $S = 3$, $t_p = 10$

Test Fold		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**All**
	Acc	**80.5**	81.9	**86.8**	81.4	85.7	84.2	**86.5**	**80.5**	**86.3**	84.3	**80.3**	84.4
none	**Pre**	86.0	84.1	91.8	88.0	89.5	89.1	91.6	85.3	91.2	89.2	84.8	89.3
	Rec	89.4	94.2	92.9	89.8	95.9	92.1	92.6	91.5	92.8	92.6	90.8	92.3
left	**Pre**	41.8	61.8	76.1	62.1	72.0	41.7	69.2	51.6	55.0	61.6	73.5	59.0
	Rec	50.5	28.4	64.3	54.3	53.3	50.3	51.9	35.1	55.6	52.9	50.4	51.4
right	**Pre**	79.7	75.2	65.0	70.0	72.7	72.7	57.3	65.8	73.4	64.4	50.2	68.2
	Rec	60.2	59.0	68.6	71.8	62.2	53.5	67.0	56.0	65.7	55.6	49.5	60.7

6.1.2.2 Maneuver-level Prediction

The previous subsection analyses the CNNs' outputs as isolated elements. This subsection evaluates complete maneuvers as a single element, considering them as continuous evolving elements. The maneuver classification procedure is the same used in 6.1.1.2. The only difference is that a few samples are added before each maneuver to provide a chance to classify them before they have started. Here we define that a maneuver is predicted only if it is detected at least 1 sample before their observed beginning. The length of maneuvers has been also normalized to 1 to avoid weighting effects. The period before the maneuver is equal for all the maneuvers so it does not need to be normalized.

Tables from 6.12 to 6.14 presents average numeric results for each trained model. Anticipation is provided in seconds to the LCE and relative to the maneuver's length, where 100% means the maneuver is correctly detected from its beginning and 0% at the LCE. However, a prediction period is added before the beginning of the maneuver extending its virtual length over the unit. If a maneuver is correctly predicted the anticipation can be higher than 100%.

Prediction is presented in two formats. As a time value which represents the average prediction time for the predicted maneuvers only, and as a percentage which shows the share of predicted lane-change maneuvers.

The AUC represents in a single number the ability to predict and detect as soon as possible all the maneuvers, including those which are not detected. The AUC value is composed as the addition of the ordinary detection AUC and the prediction AUC. The detection AUC evaluates as soon as the maneuvers are detected, considering only the samples since their observed beginning. The prediction AUC evaluates the as soon as the maneuvers are predicted, considering only the samples between the first sample of the maneuver and the beginning of the observed maneuver. Both can account for a maximum of 1 individually and together can reach a maximum value of two.

Finally, accuracy follows the same criteria than the previous subsection, specifying lane-change and lane-keeping accuracy, as well as overall training and *User Prediction Challenge* accuracy.

Table 6.12: Maneuver Anticipation vs Accuracy $S = 1, t_p = 10$

	Ant.		Pred.		AUC	Accuracy			
	[s]	[%]	[s]	[%]		LC	NLC	All	Balanced
Alexnet	3.20	106.3	0.82	63.5	1.15	78.5	68.9	70.1	75.3
GoogleNet	3.04	102.0	0.79	59.3	1.17	85.0	67.4	69.6	79.2
SqueezeNet	3.19	108.2	0.78	65.4	1.31	91.0	62.2	65.8	81.4
Resnet18	2.53	81.7	0.71	32.6	0.75	72.6	87.2	85.4	77.4
Resnet50	2.56	82.1	0.69	35.9	0.75	72.9	87.1	85.4	77.6
Resnet101	2.60	86.2	0.70	42.6	0.90	79.8	84.8	84.2	81.4

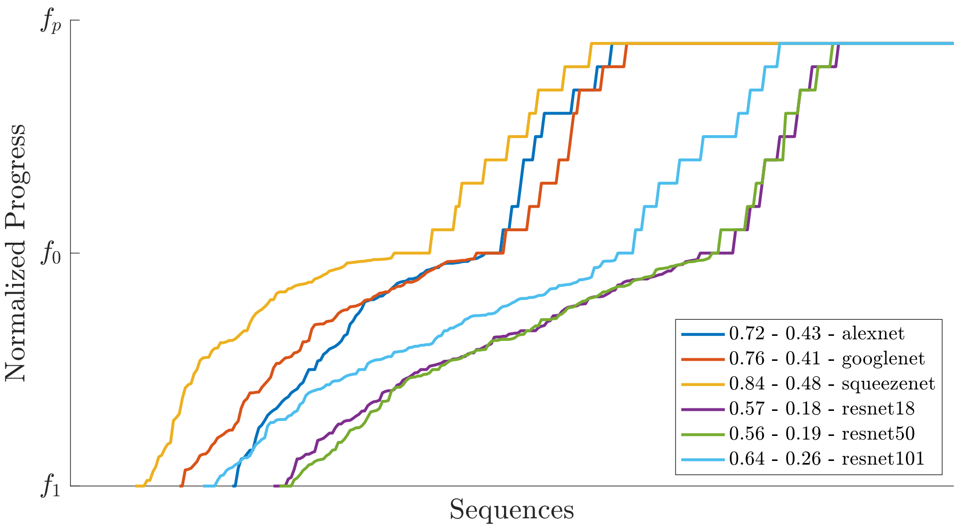


Figure 6.4: Normalized lane change detection representation. $S = 1, t_p = 10$.

Figures from 6.4 to 6.6 represents graphically the performance of the trained models detecting and predicting lane-change maneuvers. The representation is the same used in the previous subsection but predicted maneuvers are represented above f_0, which is the observable beginning of the maneuver. The top limit is f_p which points to the first sample of each maneuver. The AUC value is extracted from this representation where the rectangle created between f_0 and f_1 denotes the area for the detection AUC and the rectangle created between f_0 and f_p defines the prediction's AUC area. Models can be visually compared, better ones described most-left and most-high curves. Note the point where each curve starts defines the share of non-detected maneuvers and the point where the curve reaches f_0 defines the share of predicted maneuvers.

Table 6.13: Maneuver Anticipation vs Accuracy $S = 2$, $t_p = 10$

	Ant.		**Pred.**		**AUC**	**Accuracy**			
	[s]	[%]	[s]	[%]		LC	NLC	All	Balanced
Alexnet	2.97	106.4	0.79	65.3	1.28	86.9	67.5	69.5	80.4
GoogleNet	2.95	108.3	0.74	62.9	1.31	90.9	67.4	69.9	83.0
SqueezeNet	3.12	112.8	0.79	68.8	1.41	91.7	65.3	68.0	82.8
Resnet18	2.61	94.2	0.75	42.9	0.98	80.6	86.3	85.7	82.5
Resnet50	2.54	92.3	0.73	45.7	0.98	82.5	87.1	86.6	84.0
Resnet101	2.55	93.6	0.68	47.2	0.99	84.1	85.9	85.7	84.7

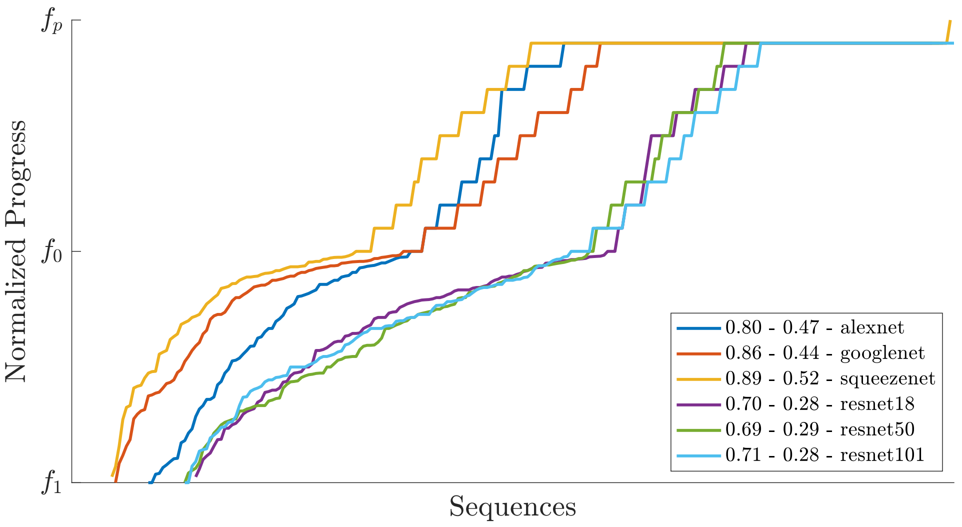


Figure 6.5: Normalized lane change detection representation. $S = 2$, $t_p = 10$.

Comparing the three tables and figures it can be observed that Alexnet, GoogleNet, and SqueezeNet models overcome widely Resnet models in predicting maneuvers as well as in detecting them. SqueezeNet's anticipation reaches 3.19 seconds, 0.71 seconds more compared with the Resnet50 model, which achieved the best single-sample classification results. The percentage of predicted maneuvers is higher for the simplest models again, 70.2% versus 49.5% at S_3 set. The average prediction time reaches 0.77 seconds for those predicted maneuvers with Alexnet and SqueezeNet models. The difference with Resnet models is narrower in this feature, achieving 0.71 seconds of prediction time on average. The AUC value is a consequence that summarizes all these causes. SquezeNet's AUC is a 32% higher than Resnet101's AUC.

Table 6.14: Maneuver Anticipation vs Accuracy $S = 3$, $t_p = 10$

	Ant.		**Pred.**		**AUC**	**Accuracy**			
	[s]	[%]	[s]	[%]		LC	NLC	All	Balanced
Alexnet	2.97	105.3	0.77	66.1	1.23	86.0	66.9	69.1	79.6
GoogleNet	3.01	106.8	0.75	65.6	1.29	90.4	67.5	70.2	82.8
SqueezeNet	3.11	110.5	0.77	70.2	1.36	90.7	67.0	69.8	82.8
Resnet18	2.58	90.3	0.69	45.3	0.95	82.7	84.8	84.5	83.4
Resnet50	2.58	90.9	0.71	45.7	0.96	82.2	84.5	84.2	83.0
Resnet101	2.69	94.9	0.72	49.5	1.03	83.5	83.3	83.3	83.4

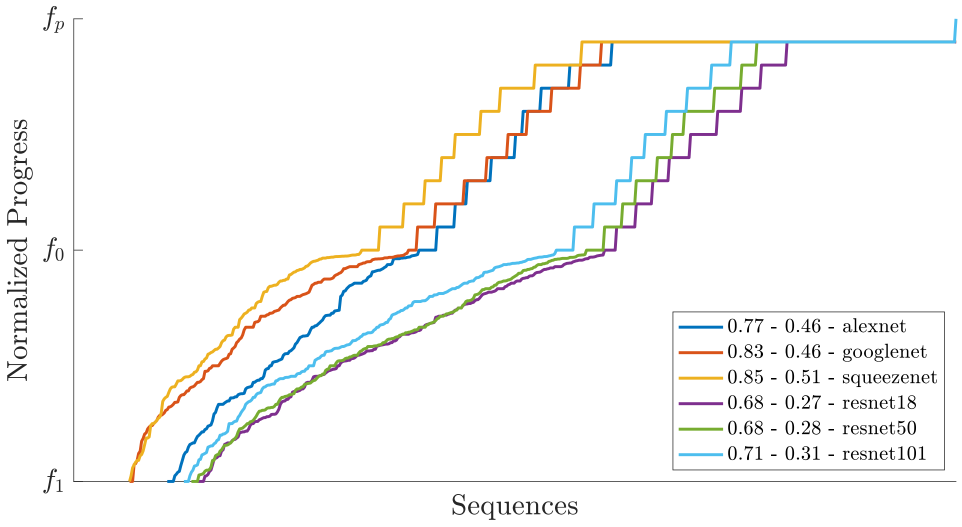


Figure 6.6: Normalized lane change detection representation. $S = 3$, $t_p = 10$.

The prediction of maneuvers produces the same effect as it was produced with the detection of maneuver regarding the accuracy. While the simplest models achieve higher accuracy for lane-change maneuvers, they are worst in detecting lane-keeping maneuvers (90.7% vs 67.0%, SqueezeNet table 6.14). On the other hand, Resnet models achieve quiet similar accuracy for lane-change and lane-keeping maneuvers (83.5% vs 83.3%, Resnet101 table 6.14). This situation is generated by the imbalance number of maneuvers in the training sets. The additional *Balance* accuracy column shows the accuracy for the *User Prediction Challenge* maneuvers, in which the amount of no lane-change maneuvers is similar to the lane change ones. In this particular set, which includes predictions, the SqueezeNet model overcomes Resnet50, which was the best detecting maneuvers (see 6.7).

Simple models anticipate more but also increase their balanced accuracy. On the other hand, Resnet models have lower anticipation periods but also lower accuracy levels. This scene is opposite to the one observed while training maneuver detections. SquezeNet model has 3.11 seconds of anticipation on average with a maneuver detection accuracy of 82.8%. Comparing detection versus prediction, the SquezeNet model has kept its LC accuracy and has reduced its NLC accuracy 5.9% to earn 0.68 seconds of anticipation.

Comparing the human and the model's ability to assess maneuvers correctly, humans reached 83.5% accuracy (see table 4.3). Alexnet, GoogleNet, and SqueezeNet models reached from 79.6% to 82.8% accuracy classifying maneuvers (see table 6.14), which is 0.7% below the human performance. The human's average anticipation was established in 1.66 seconds. SquezeNet model has average anticipation of 3.11 seconds which increases the anticipation 1.45 seconds. The Resnet models have reached performance comparable to human accuracy, but with higher anticipation periods. Resnet101 equals human's accuracy (83.4%) and increases the anticipation period of 1.03 seconds.

6.1.3 Anticipation vs Accuracy

It has been observed that the better anticipation or prediction, the lower the accuracy is. Both features are compared one by one and together with human performance in figures 6.7 and 6.8. Both figures present anticipation versus accuracy for each model together with the regression model to explain human's performance. Figure 6.7 shows the results for the detection models while figure 6.8 shows the results for the prediction models. It is easy to understand what models are better making use of this representation. The most left and up the model, the better their global performance is.

It can be observed that all the trained models are located above the human's performance line. This means that all the models have higher accuracy or anticipation than average human performance. Models Resnet50 and SqueezeNet highlight in figure 6.7, Resnet50 because the high accuracy level and the SqueezeNet because its anticipation period. SqueezeNet and Resnet101 can be considered the two best models in figure 6.8 because of their higher anticipation and accuracy levels respectively. In conclusion, the Resnet50 model trained in detecting maneuvers is the most reliable to detect maneuvers, and SquezeNet trained in predicting maneuvers is the most anticipative one. Both are comparatively better than humans in classifying and anticipating maneuvers.

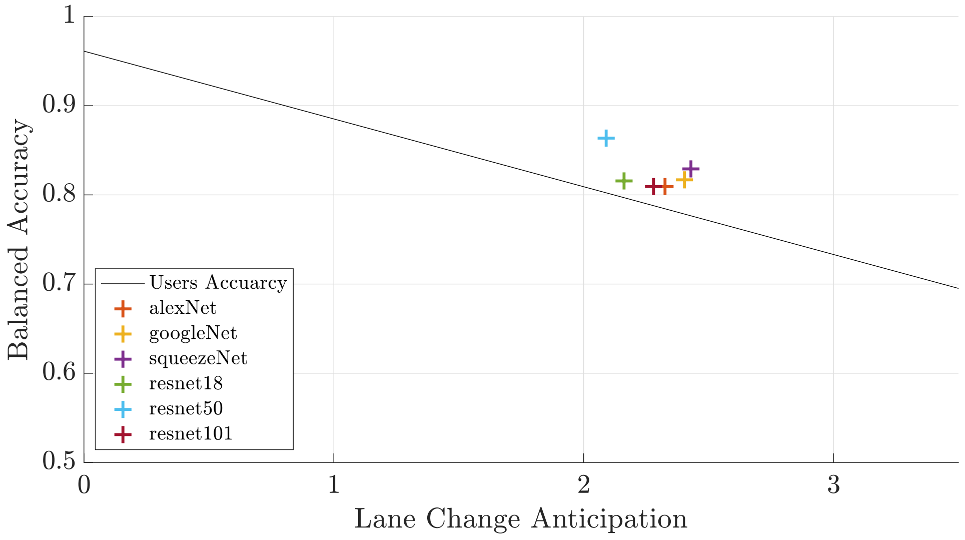


Figure 6.7: Detection models Accuracy vs Anticipation.

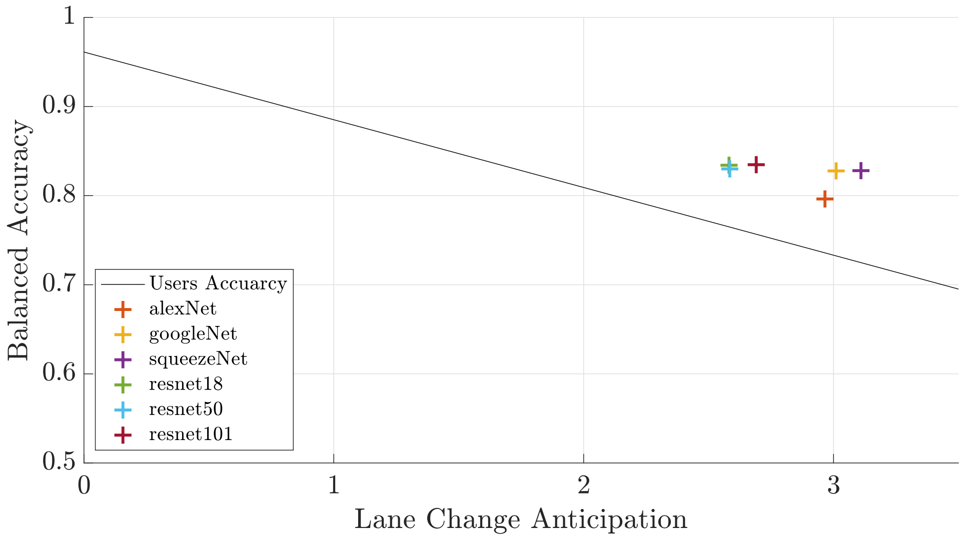


Figure 6.8: Prediction models Accuracy vs Anticipation.

6.1.4 Maneuver Prediction and Detection Visual Results

This subsection shows a set of visual results predicting and detecting lane-change maneuvers. Figures from 6.9 to 6.13 shows five sequences when lane change takes place in several different conditions. This representation shows scenes in which many vehicles are involved and all of them are classified into the three categories. Each prediction target is defined with a white cornered rectangle, this status means a lane-keeping status. The white rectangle turns red for those in which a lane change is detected by the system, additionally, a red arrow is added to highlight the sense of the lane change. Results have been generated with the Resnet101 model trained in prediction mode.

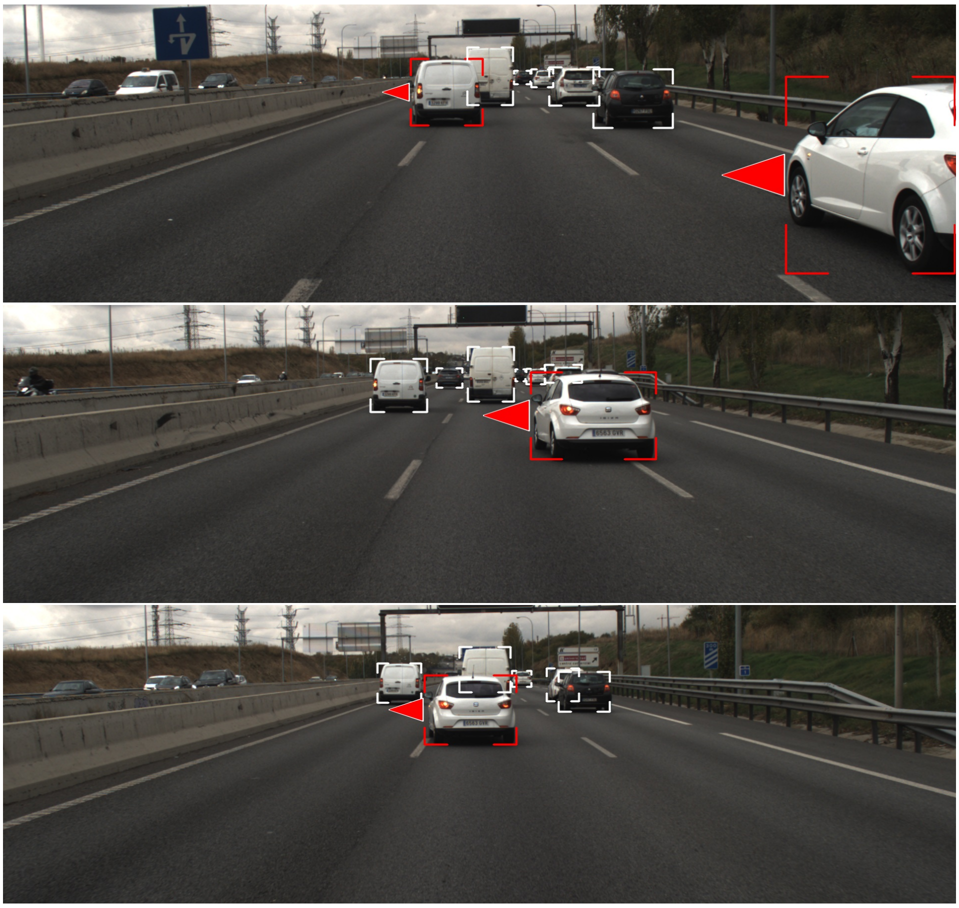


Figure 6.9: Example of lane-change maneuver detection I.

Figure 6.9 shows an evolving sequence in which two vehicles perform left lane-change maneuvers. The top image shows the first frame when the lane change is detected on the white car placed on the right lane. At the same time, a white van is performing a left lane change from the middle to the left lane. The middle and the bottom images show the evolution of the double left lane-change maneuver performed by the white car. All the samples from the beginning of the lane change to the end have been correctly classified into its corresponding category as it is shown by the red arrows. The maneuver has been anticipated 2.8 seconds before the vehicle crosses the first divisor line. Note that the white car is appearing in the image and it is only visible five frames before the lane change detection. Those vehicles which are performing lane-keeping maneuvers are also correctly categorized.

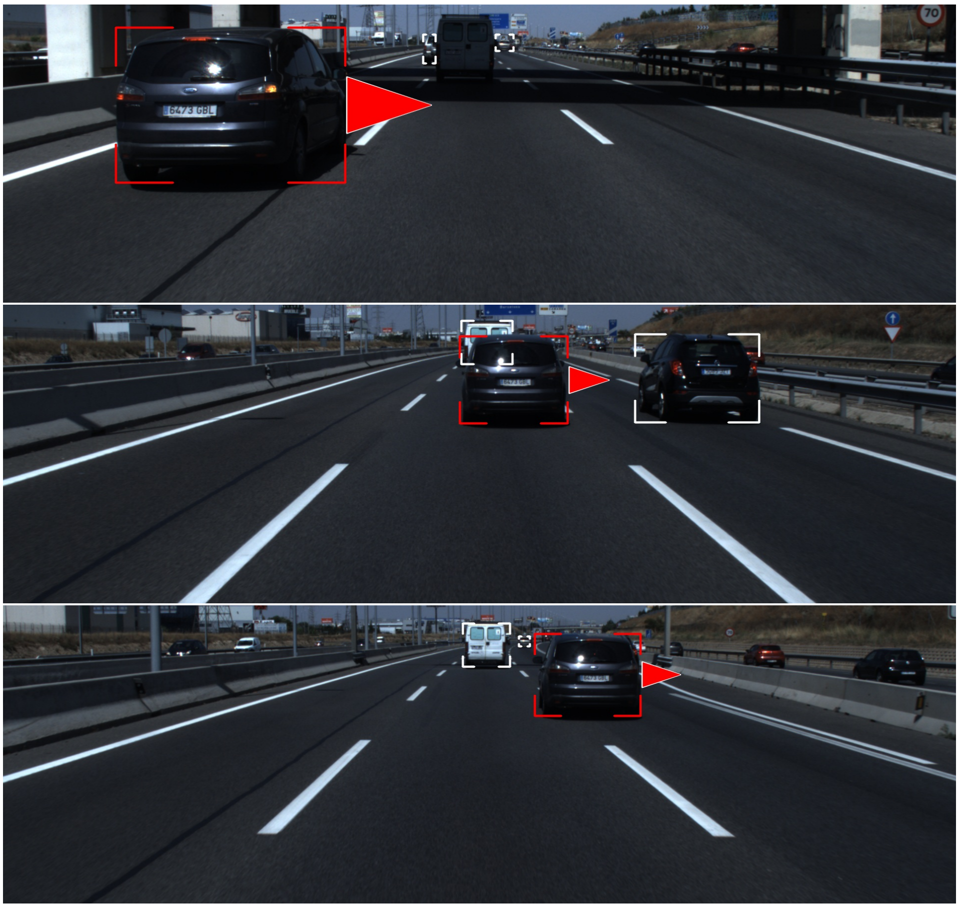


Figure 6.10: Example of lane-change maneuver detection II.

Figure 6.10 shows a double lane change again, this time from left to right. This representation shows a correct categorization of the vehicle coming from the left lane to the right one. The vehicle driving on the right lane is taking the exit and marking it with the blinker, however, the system does not detect its intention. This can be caused by a lack of movement that defines its intentions and because the exit is not visible due to the fact that the vehicle configuration occluded it. These kinds of tricky scenes are difficult to assess even for humans. The right lane-change maneuver has been anticipated 1.3 seconds, just 5 samples after the vehicle enters the camera FOV. The white van in the middle lane is performing a lane-keeping maneuver which is correctly detected as it is represented with the white rectangle.

Figure 6.11 shows an overtaking maneuver that ends in a cut-in and a cut-out maneuver when approaching to an entry ramp. In this scene, the red car in the left lane overtakes the ego vehicle and develops a right lane change merging in front of the ego vehicle. This maneuver is correctly detected with 3.1 seconds of anticipation. At the same time, the black car in the middle lane performs a cut-out maneuver which is not detected. The black car leaves the central lane because the red car enters it at a higher speed. The black car remains in the right lane until the end of the sequence. When the vehicles are approaching the entry ramp a second red car appears. The system does not detect the intention of the second car to merge the highway at the beginning of the entry ramp. This behavior can be understood from a classification point of view as a correct classification because it is not performing a lane change at that moment and there is a solid line that forbids the lane change. From a predictive point of view, there is enough information to think that the red car will join the highway through a lane change. At the same time, the black car is classified as a right lane-change maneuver. It is wrong because it will not, but the relative position of the vehicle to its lane together with the free space in front of the oncoming red car suggests that a right lane-change maneuver could take place. The black car gets on the right line at some point. The system ends up to assigning the right lane change status when the merging ramp starts to disappear. The last image in figure 6.11 shows the moment when the left lane change is detected for the second red car which anticipates the LCE 1.2 seconds.

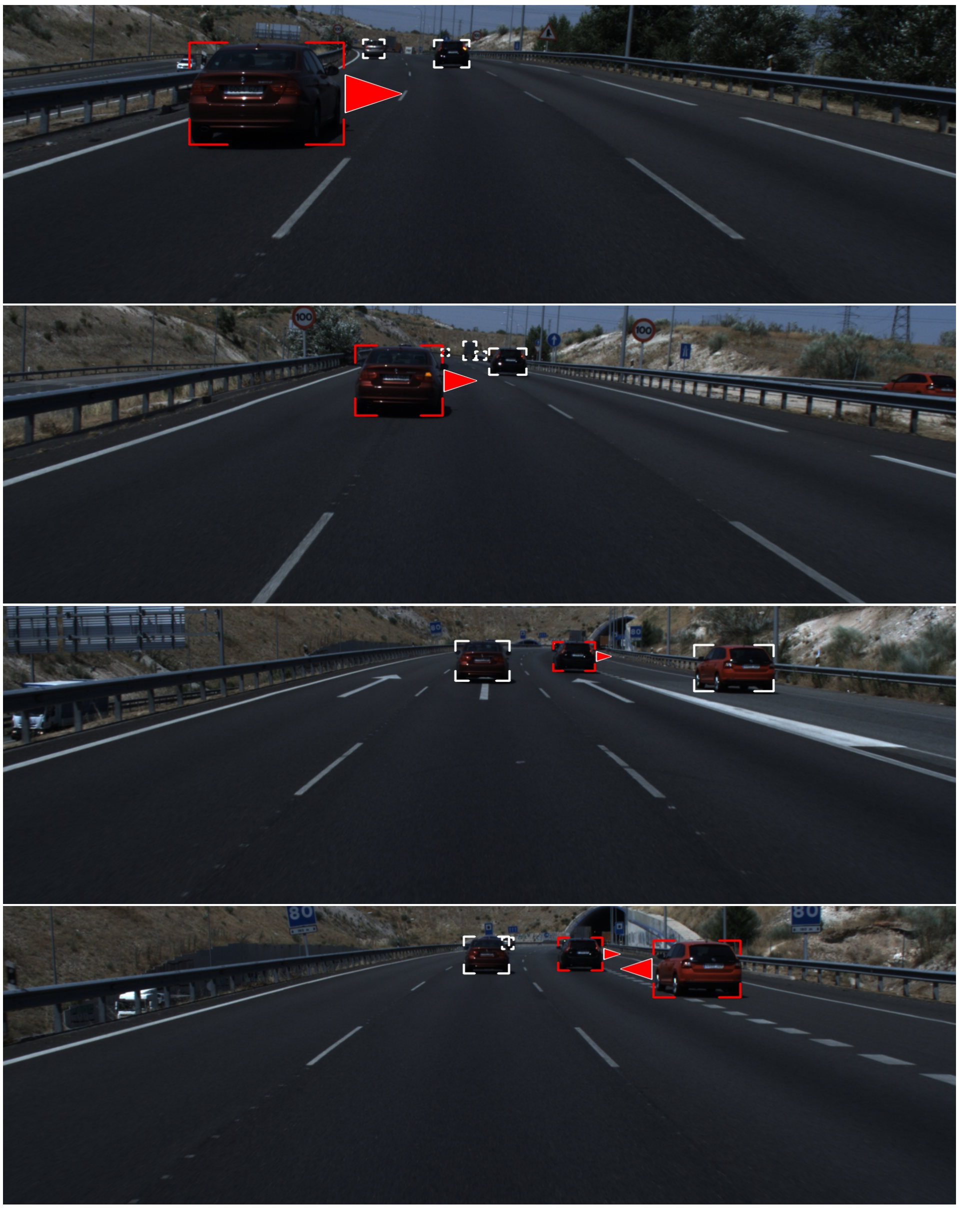


Figure 6.11: Example of lane-change maneuver detection III.

Figure 6.12 shows an example of an extreme cut-in maneuver. The black car on the middle lane (left lane with respect to the ego-vehicle) merges in front of the ego-vehicle. The free space between the ego-vehicle and the preceding vehicle is physically enough for a cut-in but unsuitable according to traffic laws and to standard driving behaviors. The top image shows a developed lane change state, however, the system does not recognize it. It can be motivated by the shortage of free space to develop the lane-change maneuver. The middle image shows the moment when the system recognizes the lane-change maneuver, 0.1 seconds before the LCE. The bottom image shows the ends of the lane change when the vehicle has stabilized in the lane and is detected in a lane-keeping status again. This behavior of the system suggests that it can judge what is more realistic based on free space and trajectories.

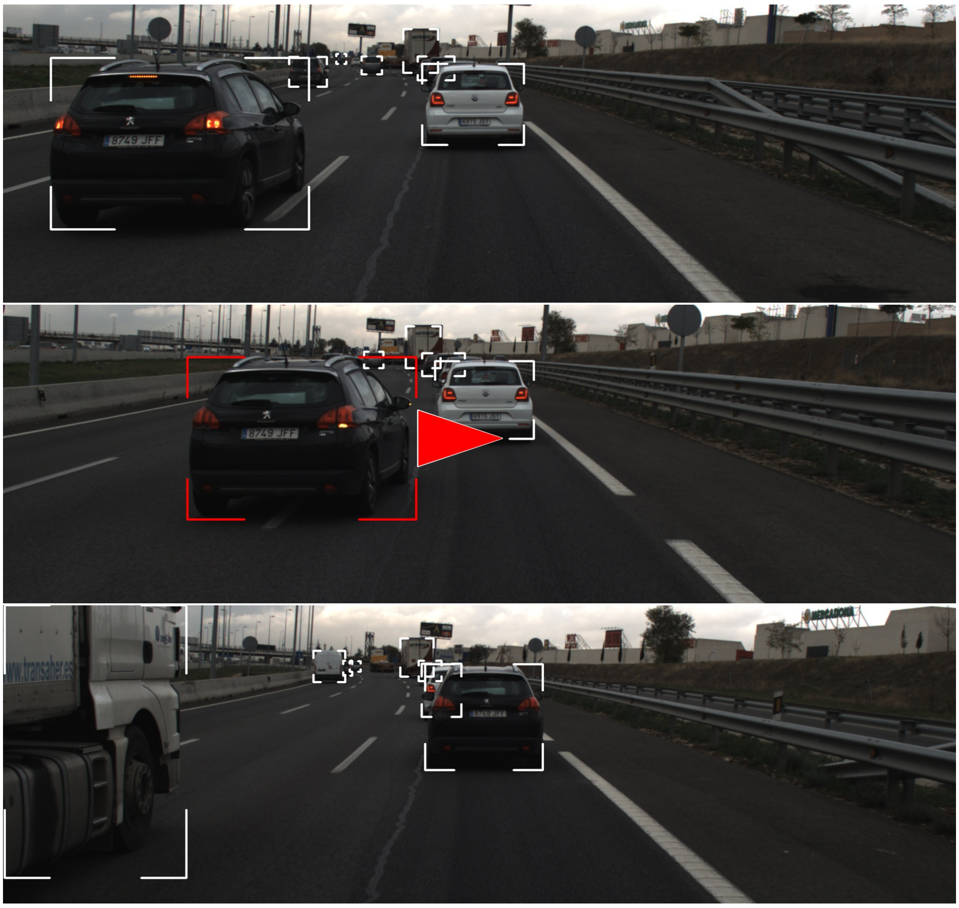


Figure 6.12: Example of lane-change maneuver detection IV.

Figure 6.13 shows the las example of lane change detections. This example shows how ego-trajectories can affect the performance of the model. In this example, the vehicle driving in the middle lane drives to reach the left lane. The top image shows the left lane-change maneuver correctly classified with 3.8 seconds of anticipation. However, 0.6 seconds after the detection of the left-lane change the ego-vehicle turns to head the left lane, and consequently, the image starts to shift to the right. The lane change of the ego-vehicle produces a distorted view of the scene that seems like a right lane change. At the moment the yaw rate of the ego-vehicle stops and the white vehicle reached the divisor line the system detects the left lane change again.

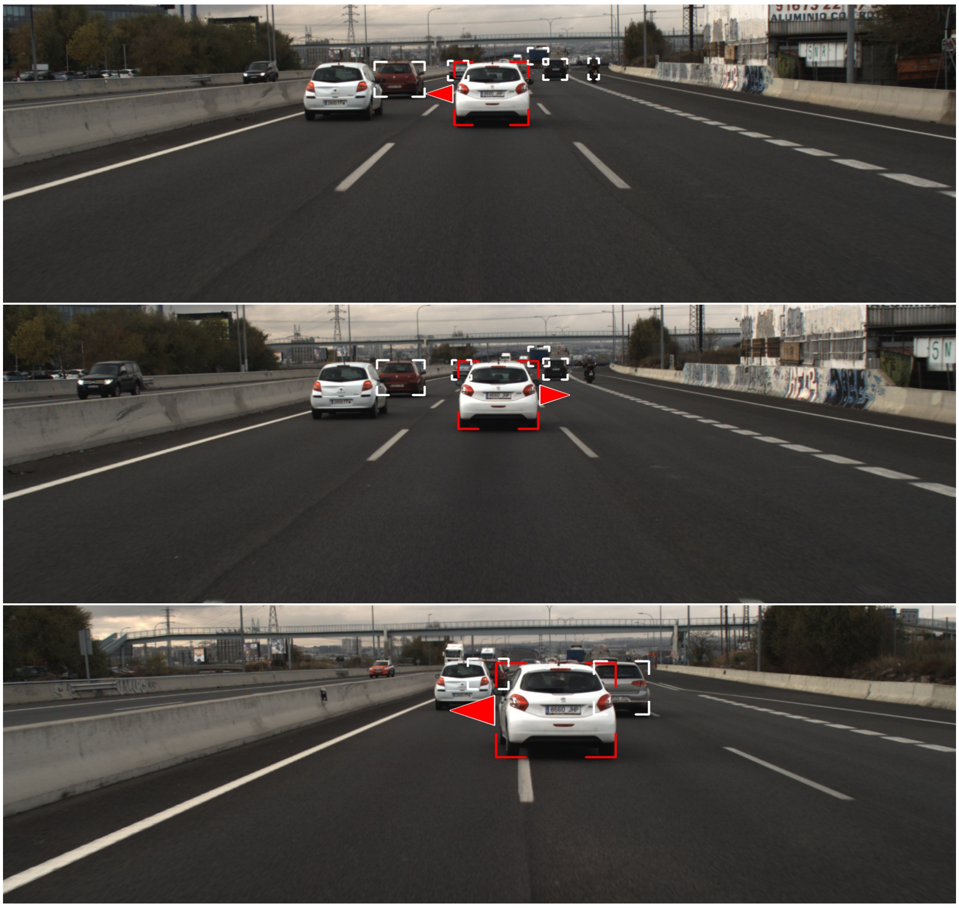


Figure 6.13: Example of lane-change maneuver detection V.

6.1.5 Applications

The models developed in this thesis have been deployed on an automated vehicle and tested in a controlled environment. AEB, ACC, and *Autonomous Emergency Steering (AES)* systems need a mechanism to take into consideration a specific surrounding vehicle. The developed models have been tested and validated in the framework of the *BRidging gaps for the adoption of Automated VEhicles (BRAVE)* European project [67]. One of the objectives of this project is to prove the validity of the advanced prediction systems.

The tests were conducted at the UTAC CERAM [68] test center owned by one of the consortium partners. UTAC CERAM is a EuroNCAP accredited agent for active safety test protocols implementation as well as for accreditation of the EuroNCAP test.

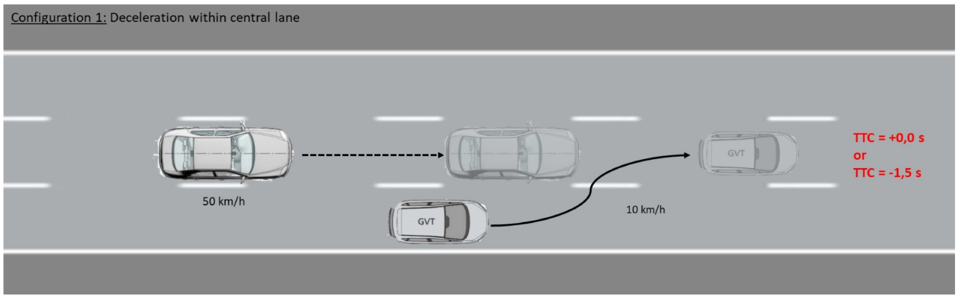
6.1.5.1 Tests Configuration

Two test configurations were tested at the UTAC CERAM center related to the scope of this thesis. Both configurations simulate an entry ramp on the highway where a vehicle will join. The trajectories of both vehicles are synchronized to generate a simultaneous arrival at the merging point. The *Vehicle Under Test (VUT)* drives at a constant speed of 50 km/h. The *Ground Vehicle Teleoperated (GVT)* drives at a constant speed of 10 km/h. These relative speeds simulate a common highway scenario in which the main traffic flow drives at 120 km/h and the side traffic flow merge at 80 km/h.

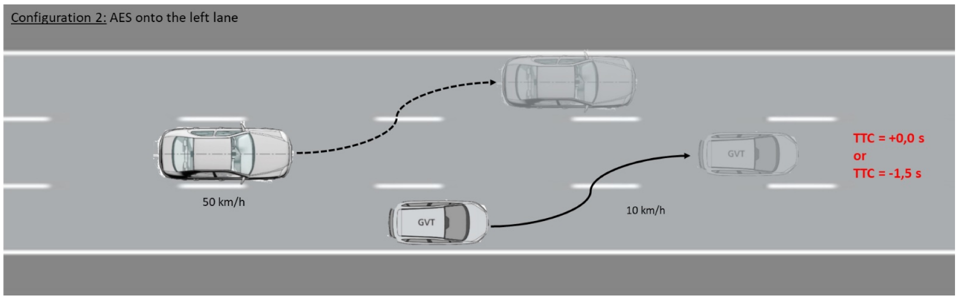
The first configuration (see figure 6.14a) evaluates the ACC functionality assuming that the GVT will merge in front of the VUT and there is no chance to use the adjoining lane. The VUT should start to brake as soon as it considers that the GVT will merge into the VUT's trajectory. The second configuration (see figure 6.14b) evaluates the AES functionality assuming that the GVT will merge in front of the VUT and the adjoining lane is available to be used. The VUT should start to change the lane as soon as it considers that the GVT will merge into the VUT's trajectory.

The VUT's performance relies on the prediction or detection of an oncoming lane change in both configurations. The earler the lane change is detected the longer the time and the distance to the GVT are. Consequently, smother actions need to be applied to control the VUT and higher the safety levels are.

Additionally, two different setups were used for each configuration.



(a) Configuration 1. ACC system test.



(b) Configuration 2. AES system test.

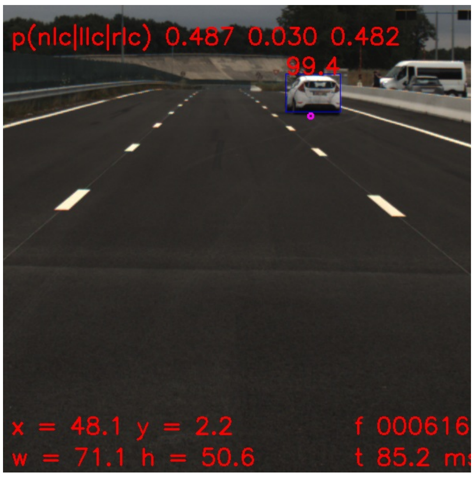
Figure 6.14: EuroNCAP test configuration.

The TTC used for the experiments were $TTC = 0$ and $TTC = -1.5$ seconds. In other words, this means that if none of the vehicles change its speed the collision point will be at the end of the GVT's merging with its rear bumper in contact with the front bumper of the VUT for $TTC = 0$ with a relative speed of 40 km/h. For the setup with $TTC = -1.5$ seconds, the relative distance from the front bumper of the VUT to the rear bumper of the GVT is -16.67 meters, which means the VUT has overtaken (or crashed) the GVT.

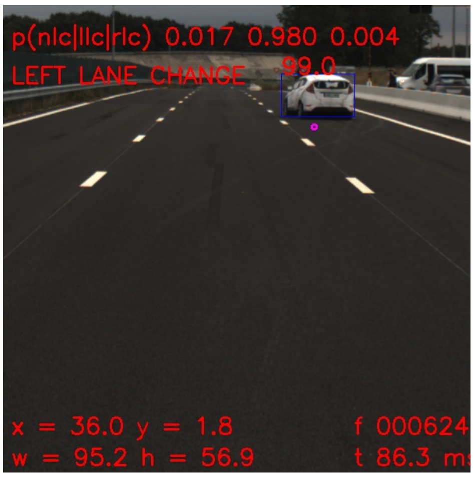
6.1.5.2 Evaluation Results and Comments

This subsection provides visual and numerical results for the described scenarios. The following figures provide images extracted from the developed interface in which relevant information can be observed such as frame number, processing time, the position of the target, and probability for each possible action. Note that the results were generated by using the Resnet50 model trained to strictly detect ongoing maneuvers because of the lack of motivation to perform lane changes by the GVT.

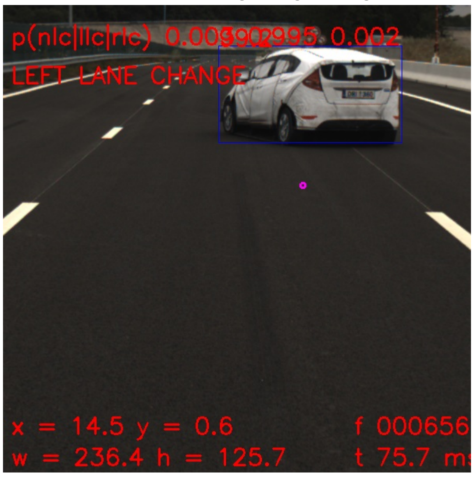
Figure 6.15 shows keyframes in one of the tests performed in configuration 1 (ACC) with $TTC = 0$. Figure 6.15a shows the first frame in which the lane change can be observed (48.1 meters). At this point, the vehicle continues at its cruising speed. Figure 6.15b shows the frame in which the system detects the lane-change maneuver and starts to perform ACC (36 meters, 6 frames, and 0.6 seconds delayed). Figure 6.15c shows which will be the traditional lane change detection, which is when the middle of the vehicle crosses the divisor lane (14.5 meters, 40 frames, and 4.0 seconds later). Finally, figure 6.15d shows the vehicle stabilized at 20 meters of distance performing ACC task.



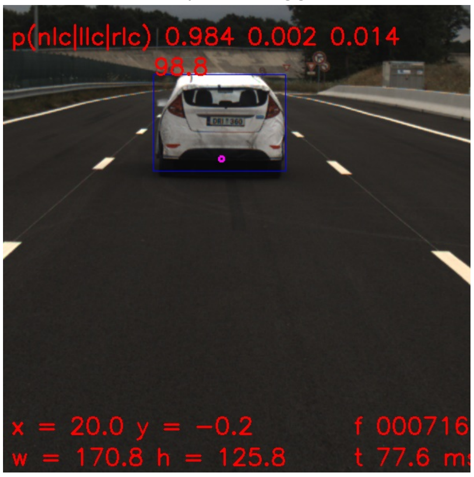
(a) Lane change beginning.



(b) System trigger.



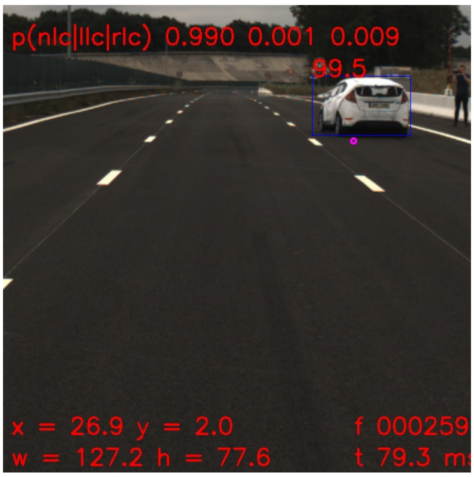
(c) Traditional trigger.



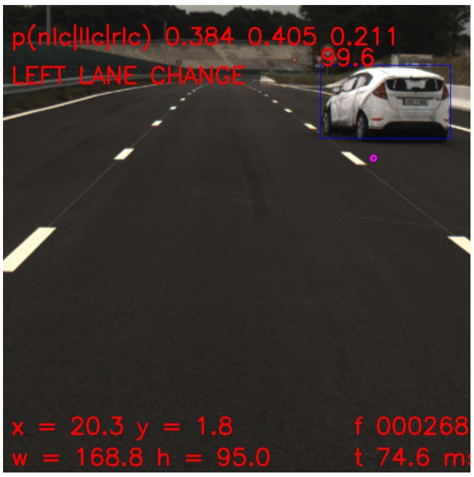
(d) Stabilized ACC.

Figure 6.15: EuroNCAP test, configuration 1, $TTC = 0$ seconds.

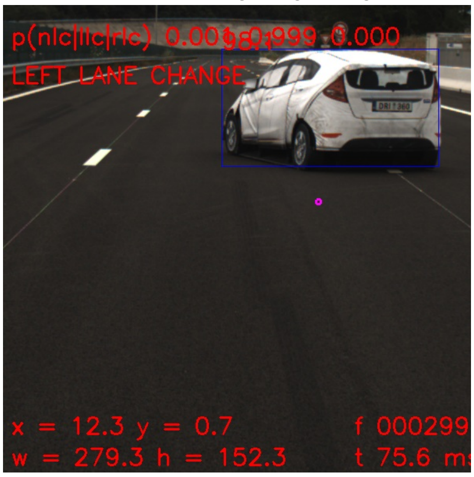
Figure 6.16 shows the same set of images for the experiment configuration 1 with $TTC = -1.5$ seconds. The difference is that the GVT starts the maneuver when both vehicles are closer. Figure 6.16a shows the GVT start to turn at 26.9 meters distance. The lane change is detected by the system at 20 meters, 9 frames, and 0.9 seconds later as it is shown in figure 6.16b. At this point, the vehicle starts to perform ACC tasks. Figure 6.16c shows the traditional lane change detection at 12.3 meters of distance, 40 frames, and 4.0 seconds later. Finally, figure 6.16d shows the vehicle stabilized at 20 meters of distance performing ACC task.



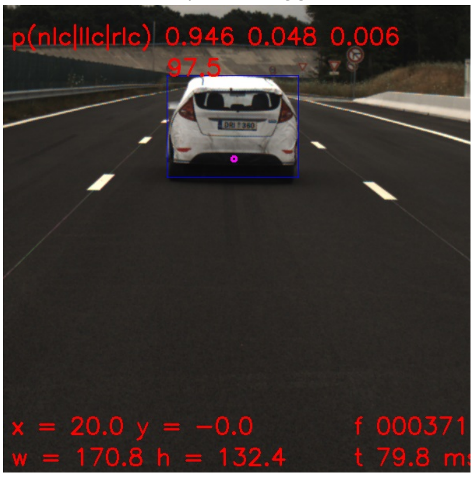
(a) Lane change beginning.



(b) System trigger.



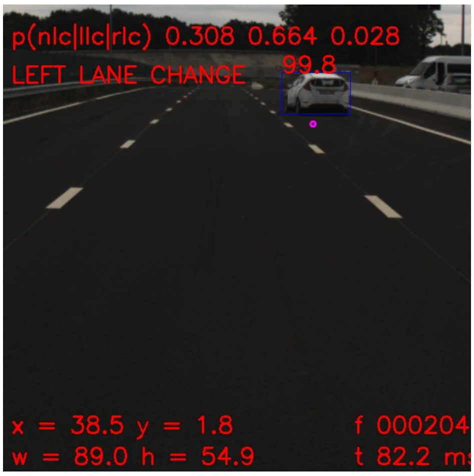
(c) Traditional trigger.



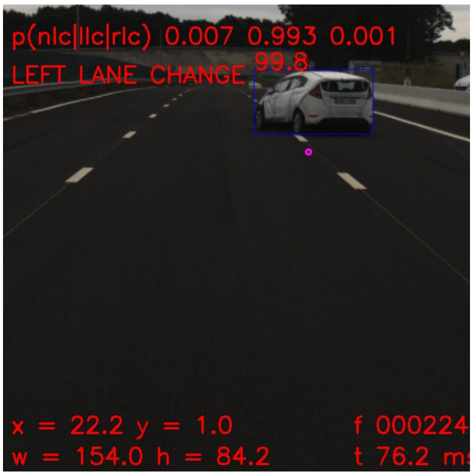
(d) Stabilized ACC.

Figure 6.16: EuroNCAP test, configuration 1, $TTC = -1.5$ seconds.

Figure 6.17 shows key frames in configuration test 2 (ACC) with $TTC = 0$ seconds. Figure 6.17a shows the frame in which the lane change is detected. This takes place at 38.5 meters to the GVT, 8 frames and 0.8 seconds after the beginning of the GVT's maneuver, at this point the AES is triggered. Figure 6.17b shows the instant in which the AES starts to modify the VUT's trajectory.

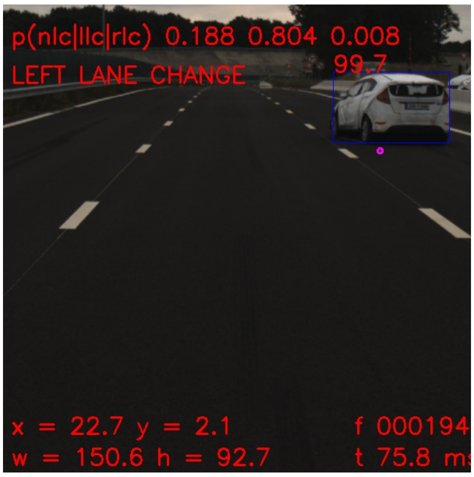


(a) System trigger.

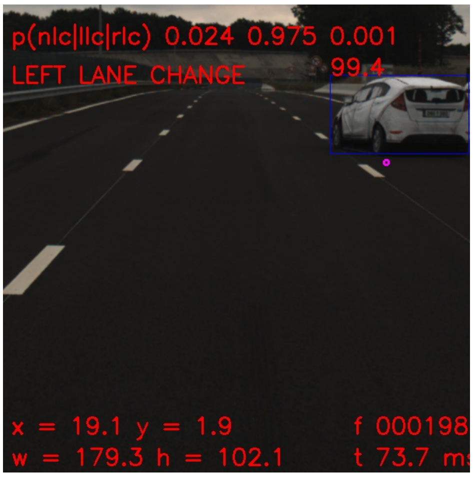


(b) AES.

Figure 6.17: EuroNCAP test, configuration 2, $TTC = 0$ seconds.



(a) System trigger.



(b) AES.

Figure 6.18: EuroNCAP test, configuration 2, $TTC = -1.5$ seconds.

Figure 6.18 shows keyframes in one of the configuration test 2 (ACC) with $TTC = -1.5$ seconds. Figure 6.18a shows the frame in which the lane change is detected. This takes place at 22.7 meters to the GVT, 9 frames, and 0.9 seconds after the beginning of the GVT's maneuver. Figure 6.17b shows the instant in which the AES starts to modify the trajectory of the VUT at 19.1 meters of distance and 4 frames later.

The independent EuroNCAP tester issued the following qualitative report comparing BRAVE and other vehicles with similar capabilities:

"The vehicle tested scored full points on the accomplished tests. EuroNCAP protocols used to obtain scores were written to assess AEB systems."

"Comparison between BRAVE vehicle and other manufacturer's vehicles with AD functions:

- *BRAVE vehicle is the best performing vehicle as far as relative distance with GVT is concerned for cut-in use cases.*
- *BRAVE vehicle was the only vehicle able to avoid any collision fully autonomously for the cut-in -1.5s TTC use case.*
- *BRAVE vehicle behaves more smoothly than other vehicles."*

6.2 Trajectory Prediction

This section presents the results generated by the proposed trajectory prediction models. This section is divided into two subsections. Subsection 6.2.1 provides quantitative results and subsection 6.2.2 presents qualitative prediction results by means of graphic representations inferred by the proposed models.

6.2.1 Quantitative Results

The quantitative results of trajectory prediction are evaluated in this subsection. For trajectory prediction evaluation, two metrics are used to evaluate the quality of each model. The RMSE and the *Mean Absolute Error (MAE)* are used as performance metrics and they are provided in both longitudinal and lateral coordinates.

The RMSE is defined in equation 6.1. Sub index k denotes each prediction step and sub index i represents each individual trajectory in a set with a total of N trajectories.

$$RMSE_k = \frac{1}{\sqrt{N}} \sum_{i=0}^{N} \sqrt{(\hat{x}_{k,i} - x_{k,i})^2} \tag{6.1}$$

The MAE represents the average error as it is defined in eq. 6.2. The RMSE weights each error by its own value, in contrast the MAE provides an average value of the error.

$$MAE_k = \frac{1}{\sqrt{N}} \sum_{i=0}^{N} |\hat{x}_{k,i} - x_{k,i}| \tag{6.2}$$

The literature evaluates commonly the performance of trajectory predictive models providing the *Average Trajectory Error (ATE)* and the *Final Trajectory Error (FTE)*. In this work, RMSE and MAE are provided for each prediction step, consequently, FTE matches with the MAE at the last prediction step. The ATE is directly computed as the MAE's average value as it is shown in eq. 6.3 where M is the total number of prediction steps.

$$ATE = \frac{1}{M} \sum_{k=0}^{M} MAE_k \tag{6.3}$$

The input block provides a kind of video clip representing 1.75 seconds of past information to generate an output block that represents the scene 2.0 seconds in advance. This output block has 8 samples at 0.25 seconds time intervals. Table 6.15 presents the RMSE for the different trained models at different prediction intervals. For clarity, some prediction intervals have been omitted, however, the prediction errors follow a linear trend. The *tanh* configuration produced unstable trainings generating divergence and ending with computation errors. These configurations have been removed from tables due to their consistent null results. U-net has been tested with 4, 5, and 6 depth levels together with the *linear* and *clippedRelu* final layers. Hereafter the U-net models will be denoted as U-net4 for the configuration with 4 depth levels and similarly for each depth level.

The first entry in table 6.15 presents results for KF which is used as baseline method for comparison purposes. The KF has been used to predict positions at the same prediction horizons as the CNN-based models.

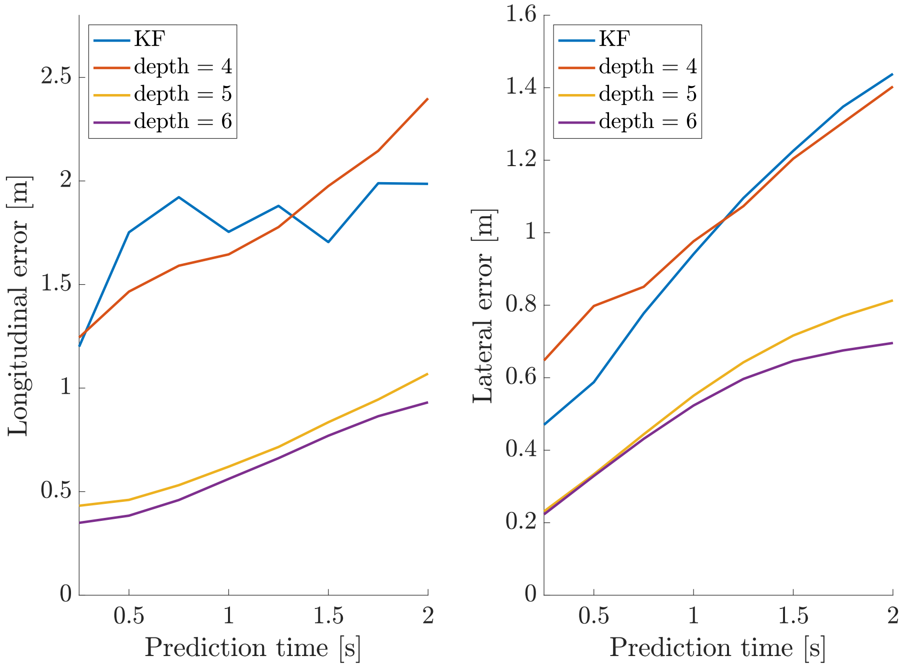
Table 6.15: Network Parameters Trajectory Prediction RMSE

	$t = 0.25$	$t = 1.0$	$t = 2.0$
Predictive model	ε_x / ε_y	ε_x / ε_y	ε_x / ε_y
KF	1.20 / 0.47	1.76 / 0.94	1.98 / 1.44
U-net4, fcn = *linear*	1.24 / 0.65	1.65 / 0.97	2.39 / 1.40
U-net4, fcn = *clippedRelu*	1.36 / 0.71	1.68 / 1.04	2.51 / 1.39
U-net5, fcn = *linear*	0.43 / 0.23	0.62 / 0.55	1.06 / 0.81
U-net5, fcn = *clippedRelu*	0.74 / 0.38	0.95 / 0.68	1.93 / 0.94
U-net6, fcn = *linear*	**0.35 / 0.22**	**0.56 / 0.52**	**0.93 / 0.69**
U-net6, fcn = *clippedRelu*	0.65 / 0.27	0.94 / 0.71	1.72 / 0.87

It can be observed that the error gets lower as the number of depth levels grows. The U-net6 produces better predictions than the U-net5, and this produces better results than the U-net4. The proportional relation between depth levels and the receptive field size and the network complexity could explain this behavior. The KF model has an RMSE comparable to the U-net4. U-net5 and U-net6 produce a lower error in both longitudinal and lateral coordinates.

Comparing the two terminal layers, there is a huge difference between models trained with the *linear* and those trained with the *clippedRelu* terminal layer. Error is nearly the half when the *linear* layer is used instead the *clippedRelu.* This difference gets smaller when less complex models with fewer depth levels are used. A priori the *clippe-*

dRelu was expected to fit better into the problem's nature. However, the nonlinearities introduced by this layer at the end of the model allows a better fitting but also introduces a flat gradient response at some points.



(a) Longitudinal RMSE. (b) Lateral RMSE.

Figure 6.19: Trajectory prediction. RMSE.

Figure 6.19 shows the RMSE for both longitudinal and lateral error as a visual complement of the values detailed in table 6.15. *ClippedRelu* configurations have been omitted due to their bad performance in comparison with the base line. As it was said before the best model is the U-net6 with the *linear* terminal layer followed closely by the U-net5 with the same configuration.

It is observed that KF's errors are similar to the simplest U-net4. It can be explained because this U-net has a small receptive field and the future positions of the vehicles are inferred based only on near objects which basically is the vehicle itself. U-net5 or U-net6 increases exponentially their receptive fields including inter-vehicle interactions. This is the reason why they can perform better predictions than the KF and the U-net4.

U-net6 prediction errors are 53% and 52% lower with respect to the baseline method in longitudinal and lateral coordinates at 2.0 seconds prediction horizon. The U-net5 prediction errors are 46% and 44% lower for the longitudinal and lateral errors respectively.

Table 6.16 shows the MAE for the *linear* set of network configurations. As an unweighted metric it is easier to understand expected errors at each prediction time. Moreover, ATE and FTE are provided as common literature metrics.

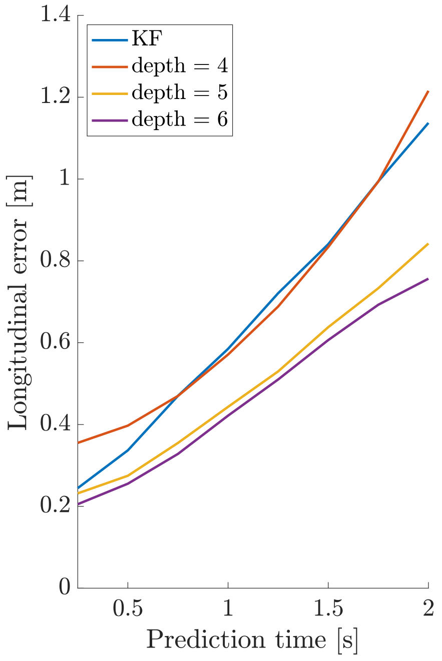
Table 6.16: Network Parameters Trajectory Prediction MAE

			FTE	ATE
	$t = 0.25$	$t = 1.0$	$t = 2.0$	
Predictive model	ε_x / ε_y	ε_x / ε_y	ε_x / ε_y	ε_x / ε_y
KF	0.24 / 0.22	0.58 / 0.65	1.13 / 1.04	0.67/0.68
U-net4, fcn = *linear*	0.35 / 0.18	0.57 / 0.46	1.43 / 0.85	0.69/0.49
U-net5, fcn = *linear*	0.23 / 0.15	0.44 / 0.39	0.84 / 0.59	0.51/0.40
U-net6, fcn = *linear*	**0.20 / 0.14**	**0.42 / 0.38**	**0.76 / 0.53**	**0.47/0.38**

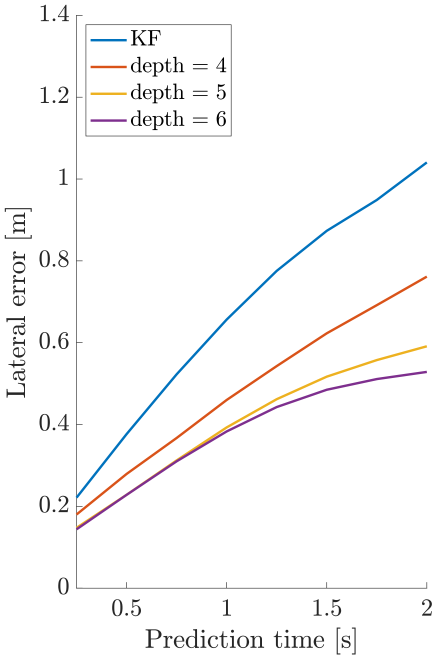
MAE values have been notably reduced compared with the RMSE values. However, the RMSE arguments are valid for the MAE. As deeper the network the better the predictions are. KF's errors are comparable to the U-net4 model again. As remarkable points, the ATE for the U-net6 is 0.51 and 0.40 meter for lateral and longitudinal errors respectively. This means that the longitudinal and lateral errors are between 0.51 and 0.40 meters on average. The FTE for the U-net6 reaches 0.76 and 0.53 meters in longitudinal and lateral errors on average for a 2.0 seconds prediction horizon.

Figure 6.20 shows the MAE for both longitudinal and lateral error as a complement of table 6.16. It can be observed that the U-net4 has a similar longitudinal error but a lower lateral error than the KF. A narrower lateral error distribution can produce a small difference between RMSE and MAE values.

The U-net6 prediction errors are 33% and 49% lower in longitudinal and lateral with respect to the KF model at MAE level for 2.0 seconds prediction horizon. If the ATE is used to compare models, the U-net6's ATEs are 30% and 44% lower than the KF's ATEs for longitudinal and lateral coordinates respectively.



(a) Longitudinal MAE.



(b) Lateral MAE.

Figure 6.20: Trajectory prediction. MAE.

These results were achieved using plain Gaussian representation of vehicles. Now, the best U-net configuration is tested using Gaussian and rectangle representations together with and without lanes. Table 6.17 shows the MAE, FTE and ATE for these configurations.

Table 6.17: Input Configurations Trajectory Prediction MAE

			FTE	ATE
	$t = 0.25$	$t = 1.0$	$t = 2.0$	
Input Configuration	ε_x / ε_y	ε_x / ε_y	ε_x / ε_y	ε_x / ε_y
Gaussian	0.20 / 0.14	0.42 / 0.38	0.76 / 0.53	0.47/0.38
Rectangle	0.26 / 0.14	0.48 / 0.39	1.04 / 0.59	0.53/0.41
Gaussian + lanes	0.19 / 0.16	0.43 / 0.40	0.75 / 0.56	0.48/0.41
Rectangle + lanes	0.24 / 0.15	0.49 / 0.37	1.07 / 0.55	0.53/0.40

It is observed that the Gaussian representation of vehicles helps to increase the performance of the prediction system compared to the representation based on rectangles. This fact can be explained because the gaussian representation can model the probability of the vehicle to be using a specific area in the scene. The area near the actual position of the vehicle has a small influence. In contrast, the rectangle representation is a kind of binary input, where areas denoted as vehicles influence on the output but those areas surrounding the vehicle cannot. The Gaussian representation seems more robust.

The representation or not of the lanes has no effect on the prediction performance. It was expected to increase the prediction performance by representing lane information in the input blocks, however, no significant changes have been observed. Lane markings are represented by single-pixel lines in order to avoid occluding vehicle positions. This fact may be the cause to produce no changes.

In conclusion, the U-net model trained with 6 depth levels and the *linear* layer as the terminal layer using the vehicle's Gaussian representation produces the lowest errors from all the trained models. This configuration overcomes in both longitudinal and lateral coordinates the KF model which has been used as baseline. U-net with 7 or more depth levels is expected to produce better results due to the observed trend, however, it is impossible to be trained with the currently available hardware.

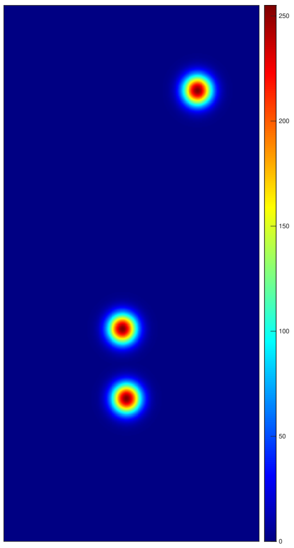
6.2.2 Qualitative Results

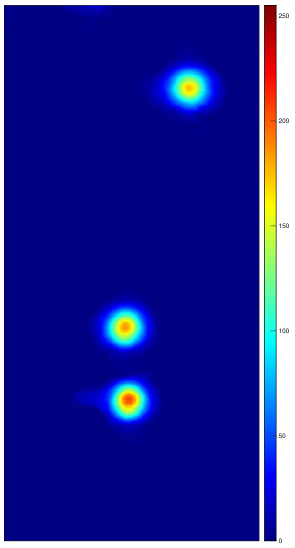
This subsection provides qualitative results and examples of the scenes predicted using the U-net architecture.

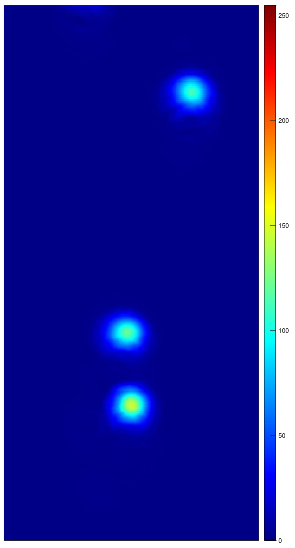
Figure 6.21 shows a scene predicted using the different network configuration and the input representation exposed above. In general, the higher the peak values and rounded the output objects the better the predictions.

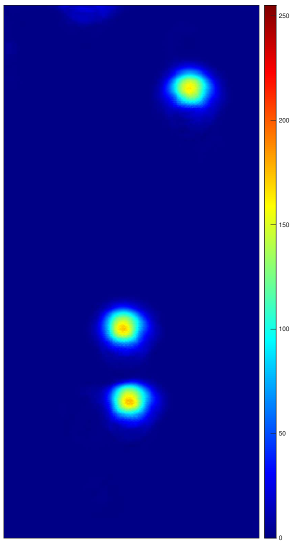
Figure 6.21a shows the desired output for the analysis sample. The positions of each vehicle are denoted by a peak value that reaches 255 intensity value. This is the output that the predictive model must generate based on a specific input. Figure 6.21b shows the generated output of the U-net6 with the *linear* terminal layer and using Gaussians to represent vehicles in the input block. Compared with the other figures this is the most likely desired output, and numerically it is, as it was exposed in tables 6.17 and 6.16. The following images show how the system's outputs degrade when worse configurations are used.

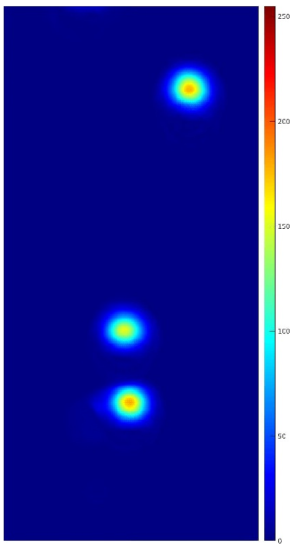
Figure 6.21c shows the effect of the *clippedRelu* terminal layer. The prediction seems quite similar for both terminal layers, however, the image loses definition, and positions extracted with the *linear* layer are two times more precise. Note that one pixel is equal to 0.2 meters in longitudinal and 0.1 meters in lateral.


(a) Desired output.


(b) *Linear* layer + Gaussian representation.


(c) *ClippedRelu* layer + Gaussian representation.


(d) *Linear* layer + Rectangles representation.


(e) *Linear* layer + Gaussian + Lanes representation.

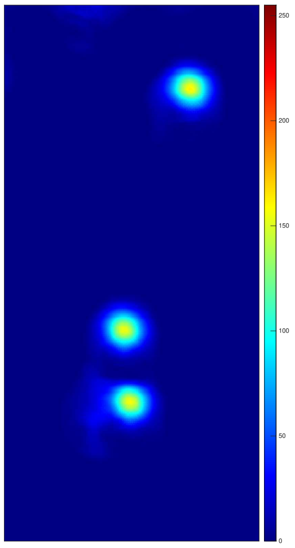
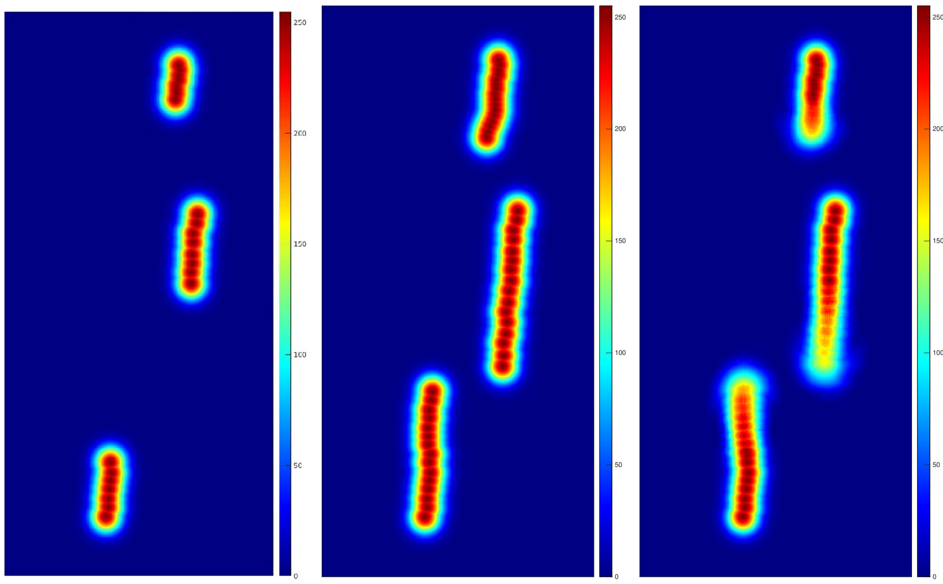

(f) *Linear* layer + Rectangles + Lanes representation.

Figure 6.21: Network configuration and Input representation's effect on output images.

Figure 6.21d shows difference in the input representation using rectangles instead Gaussians. The output image has lower definitions but still performs quite well in comparison with the *clippedRelu* variation. Figure 6.21e shows the output but now including lanes in the input representation. The definition is better when using rectangles but still lower than the plain gaussian representation. Finally, figure 6.21f shows the generated output using rectangles and lanes in the input block. As it happens with the Gaussian representation the use of lanes does not provide any improvement.

Figure 6.22 shows an example of a sequence prediction. The images shown in this figure are cumulative past future samples on a heatmap representation. The trajectory of each vehicle is represented as a kind of worm where the first half represents past positions and the second one the future positions. Figure 6.22a shows the first part of the trajectory which corresponds to the input data used to predict the trajectories. Figure 6.22b shows the input data representation together with the expected output data, this is the ground truth. Figure 6.22c shows the input and the predicted positions. Predictions become less defined for longer prediction periods.

Note that this particular sequence shows three vehicles driving at different speeds compared with the ego-vehicle, it is the only way to show this kind of representation. Otherwise, consecutive vehicle positions would be stacked in a small area and trajectories cannot be appreciated.



(a) Input sequence. (b) Input and output sequence. (c) Input and predicted sequence.

Figure 6.22: Trajectory prediction example. Time series visualization.

Figure 6.23 shows a detailed view of two trajectories corresponding with the sequence above. The two left images are a long trail sequence in which the FTE is 0.22 and 0.24 meters for longitudinal and lateral directions respectively. The predictions become less defined with higher prediction horizons. The two right images are a short displacement trajectory ended with a non-expected lateral displacement. As far as there is no evidence of lateral displacement in the input sequence and there are no other agents that could motivate the lateral displacement it is not expected to predict it. Therefore, the FTE reaches 1.5 and 0.48 meters for longitudinal and lateral coordinates respectively.

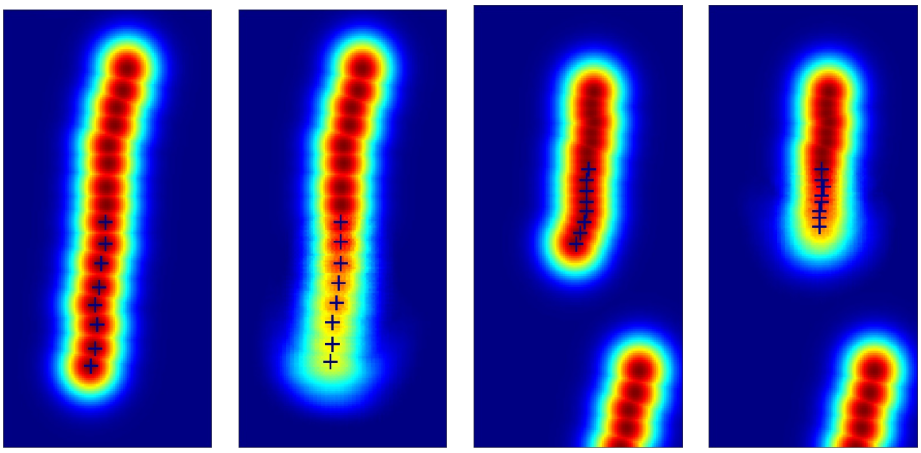
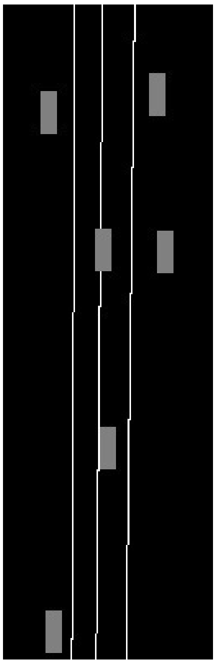
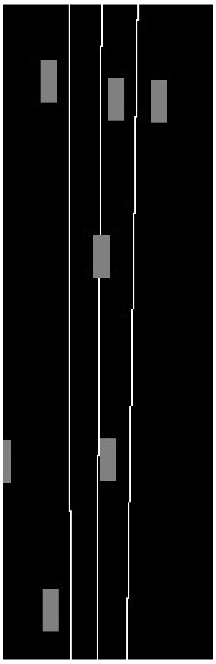
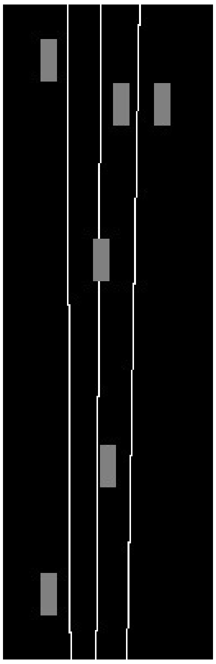

(a) Traj. 1 GT. (b) Traj. 1 Pred.. (c) Traj. 2 GT. (d) Traj. 2 Pred.

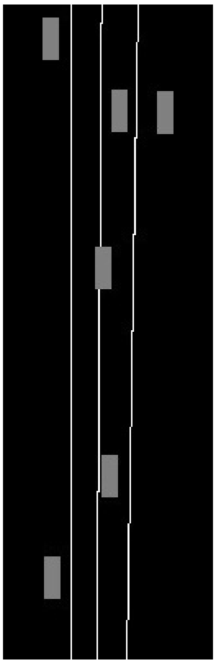
Figure 6.23: Detail of trajectory prediction.

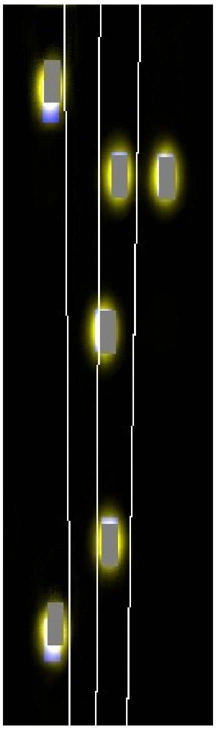
Finally, figure 6.24 shows a sort of samples in a sequence, using both input and output representation to understand the scene as better as possible. Note that lanes have been used in the predictions only for representation purposes. The actual position of the vehicle is represented with a gray rectangle in all images. The last known position of each vehicle is represented with a blue rectangle in the prediction images. Predicted positions of vehicles are denoted by the yellow Gaussian distributions. The center of each vehicle corresponds with the mass-center of each Gaussian. Images have been converted back to its natural aspect ratio. Top images have been cropped due to the lack of information in the deleted areas to fit in a single page representation.

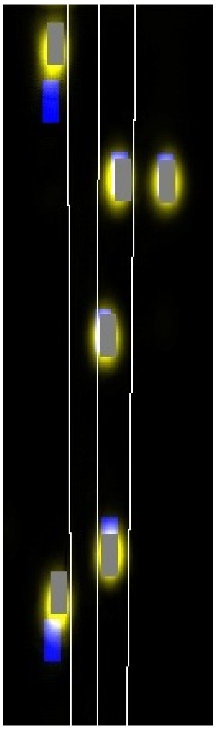

(a) $t = -1.75$.

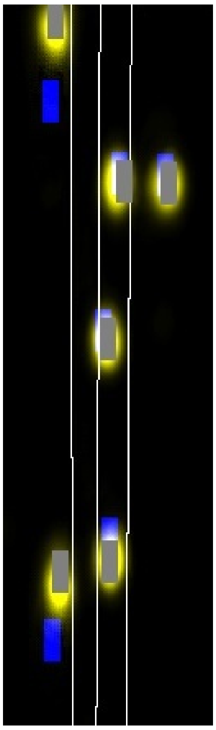

(b) $t = -1.0$.


(c) $t = -0.5$.


(d) $t = 0.0$.


(e) $t = 0.25$.


(f) $t = 1.0$.


(g) $t = 1.5$.

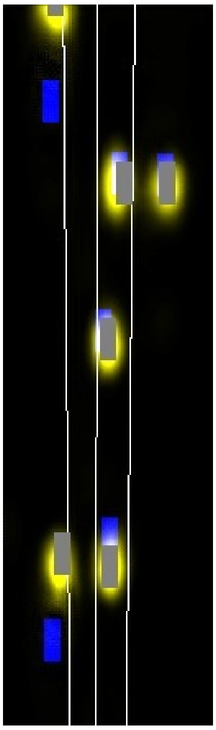

(h) $t = 2.0$.

Figure 6.24: Final Trajectory Prediction Results.

6.2.3 Application to Static Sensor Data

The proposed architecture has been trained using the PREVENTION dataset which includes data recorded with onboard sensors. However, due to the extended use of static sensor data this training has been replicated using the HigHD dataset. This dataset included static imagery taken from a drone over the study area. The study area covers only a straight stretch of highway with 3 or more lanes. This data contains two traffic streams at the same time.

The resolution used to represent the input data has been modified to fit in the GPU size. The original data frame rate was 25 Hz, it has been reduced close to 4 Hz, removing five of each six samples.

More than 28K trajectories and 93K samples were used to train the model. The test set has around 7K trajectories with 25K samples. The results are presented in table 6.18 as an MAE for both longitudinal and lateral.

Table 6.18: Trajectory Prediction MAE. HigHD Dataset.

			FTE	ATE
	$t = 0.25$	$t = 1.0$	$t = 2.0$	
Predictive model	ε_x / ε_y	ε_x / ε_y	ε_x / ε_y	ε_x / ε_y
U-net5, fcn = *linear*	0.65 / 0.21	1.17 / 0.27	1.57 / 0.36	0.97 / 0.22
U-net6, fcn = *linear*	**0.29 / 0.01**	**0.53 / 0.02**	**0.82 / 0.04**	**0.51 / 0.02**
U-net5, fcn = *clippedRelu*	0.92 / 0.47	1.35 / 0.52	1.67 / 0.63	1.03 / 0.39
U-net6, fcn = *clippedRelu*	0.57 / 0.27	1.01 / 0.33	1.37 / 0.41	0.85 / 0.25

Note that the results are in the same line than those achieved with the PREVENTION dataset. The best configuration is the U-net6 using the *linear* terminal layer. The longitudinal error is quite similar compared with the PREVENTION data. However, the lateral error is extremely low. This suggests that there are fewer lateral displacements in the HigHD dataset. The longitudinal FOV in the HigHD dataset is five times longer, and the lateral almost the same.

Figure 6.25 shows a sequence extracted from the HigHD dataset. In this figure, some samples of the input and output block are presented. Four top images represent an input sequence data, four bottom represents predictions at $t = \{0.25, 1.0, 1.5, 2.0\}$ seconds ahead. Red crosses represent the position of the prediction targets, and green plus symbols are the predicted positions extracted from the image generated by the U-net model.

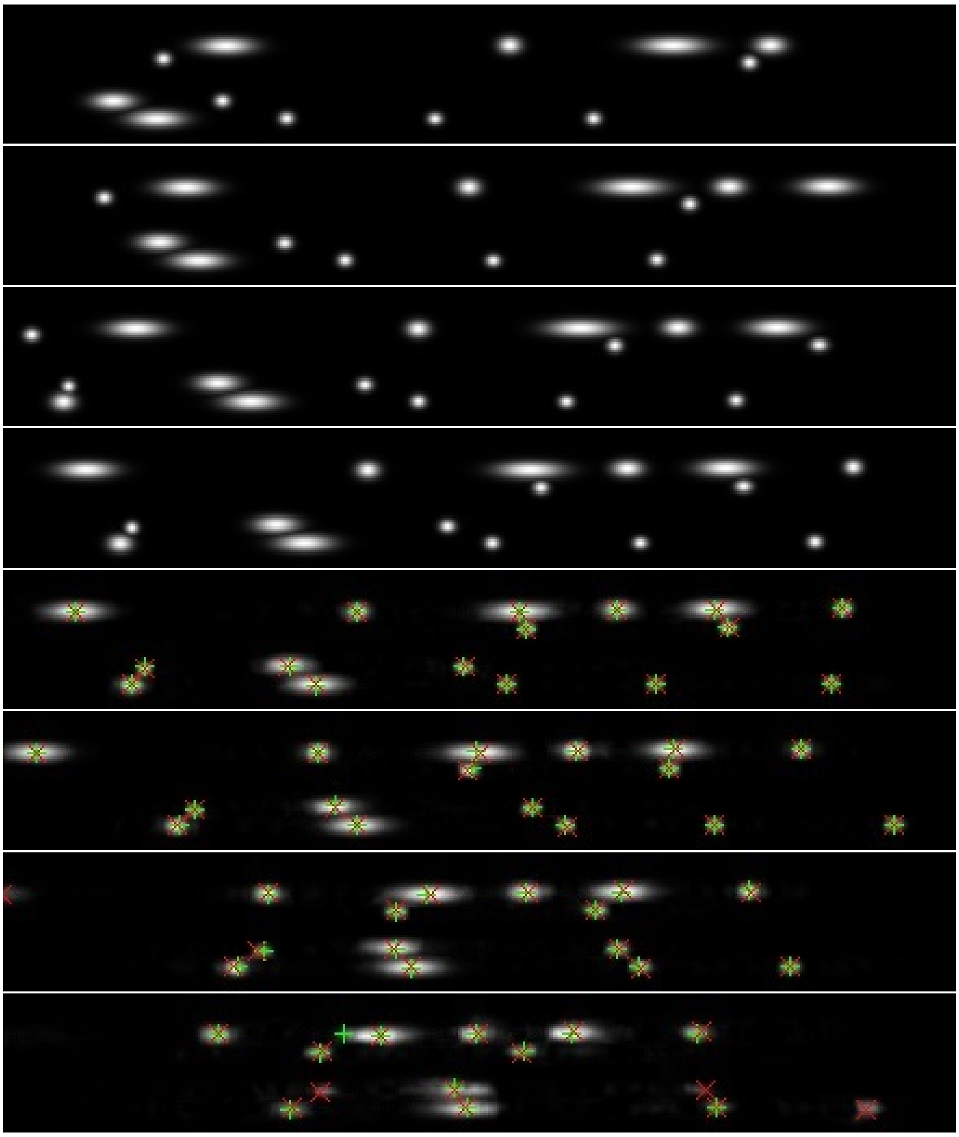


Figure 6.25: Data representation in a BEV of predicted vehicles using HigHD data.

6.3 Conclusions

In conclusion, two types of predictive models have been tested and evaluated, one to detect and predict lane-change maneuvers and other to predict vehicle trajectories.

The maneuver prediction system has proved to have better accuracy and anticipation ratio than humans by means of comparison with the *User Prediction Challenge*. The Resnet50 model is the best choice to detect maneuver and overcomes human's accuracy by 2.5% and anticipation by 0.43 seconds. Model Resnet101 equals human accuracy but it extends the anticipation period 1,03 seconds. These systems can anticipate lane changes up to 3.11 seconds on average with an accuracy ratio up to 86.4%. The detection maneuver model has been tested under controlled conditions as an ACC and AES systems trigger and it showed an outstanding performance compared with traditional decision-making algorithms.

The trajectory prediction system takes advantage of graphic representations to infer future vehicle positions by means of the U-net model. U-net model trained with 6 depth levels and using Gaussian representations of vehicles has shown prediction errors up to 50% lower than the KF. Results suggest that deeper networks can achieve better prediction performance. This system has been trained using onboard sensor data as well as static sensor data. Results show that prediction from onboard sensors is more complex than from static points of view.

Chapter 7

Conclusions and Future Work

This chapter presents global conclusions and discusses the main contributions developed in this thesis as well as the future lines of research that can be followed.

7.1 Conclusions

The goal of this thesis was the development of predictive systems to infer future intentions and trajectories of vehicles in highway scenarios.

- It was found that available public datasets were not specifically built to develop trajectory nor intention predictive models. Those who can be used to develop trajectory prediction models are based on static recording platforms. The PREVENTION dataset has been specifically built to fulfill this void. The models developed in this thesis have been trained and tested using this dataset.
- The *User Prediction Challenge* has proved that humans react to ongoing lane changes regularly instead of predicting them. The study shows that users only predict 13% of lane-change maneuvers in advance. The percentage of correctly detected lane changes before the vehicle crosses the lane rises to 85%. The average user delays lane-change detection 1.08 seconds.
- The developed maneuver prediction system has overcome human's accuracy by 2.5% and anticipation by 0.43 seconds.
- The maneuver prediction system has been tested as ACC and AES trigger mechanism, showing an improvement of 2.6 seconds compared with traditional decision-making algorithms.

- The trajectory prediction systems have been developed using the U-net model and the PREVENTION dataset. It has shown a better performance compared with a KF baseline method. The developed model has shown prediction errors up to 50% lower than the baseline method.
- Alternatively, this model thought and designed to fit in the PREVENTION dataset features has been used to predict trajectories with a static recorded dataset exhibiting better results. This proves the versatility and adaptability of the proposed approach.

7.2 Main Contributions

After the review of the state of the art and considering the discussion presented before, the main contributions of this thesis are:

- Development of the PREVENTION dataset. An onboard sensor dataset to provide specifically designed data for trajectory and lane change prediction models. This free-access dataset is open to the scientific community and provides more than 4 million vehicle detections, 3000 trajectories, and 900 lane changes in close to 6 hours of recordings.
- A social study has been conducted to evaluate human performance to predict lane changes. This study evaluates the PREVENTION scenes to set a baseline for further comparisons.
- Two novel CNN based prediction models are presented:
  - The maneuver prediction approach is based on a CNN model to classify enriched images. Context information, vehicle interaction, motion histories, and a target selection method are efficiently encoded in an enriched image to detect and predict lane changes in highway scenarios. This kind of representation has the advantage of being virtually unlimited in the number of vehicles in the scene and the number of past vehicle representations. In contrast with 4D image inputs, the data size does not increase with the number of temporal instances. In contrast with the state of the art approaches, this novel image-based proposal is not limited to a fixed number of vehicles or a fixed vehicle distribution. Besides, none of the existing works uses the appearance of the image to predict or classify the maneuvers of the surrounding vehicles.

- The trajectory prediction approach is based on a U-net model and uses vehicle graphic representations into a BEV to pattern interactions. The model uses vehicle detections and lane information to create a virtual representation of the scene, but it is not limited to these features. All kind of information can be easily included and the model is virtually unlimited in the number of considered vehicles. Moreover, the trajectory prediction system uses one single prediction step to generate all the future positions for all the vehicles, unlike most of the works in the state of the art.

7.3 Future work

After the review of the state of the art and based on results and conclusions derived from this work, several research lines can be followed to improve the performance of the systems or either take advantage of these systems in other applications.

- Machine learning approaches need tons of data to reach their maximum potential results. The first obvious future work is the improvement of the PREVENTION dataset by recording more data. The value of the dataset can be measured by the richness of the data but also by its volume. However, this is a monotonous resource and time-consuming task. The setup of a prediction challenge is a basic step in the improvement of the dataset. This will motivate researches to use it and publish their results.

- The *User Prediction Challenge* has shown important results relative to maneuver prediction. However, the system is ready to use the PREVENTION dataset to evaluate other human parameters such as attention, focus, fatigue, and others by using external cameras.

- The maneuver prediction system has shown a good performance detecting and predicting lane-change maneuvers in highway scenarios from an onboard sensor point of view. It could be interesting to exploit this approach to predict maneuvers on intersections from a static point of view such as infrastructure cameras. This prediction model could help to manage the traffic at controlled intersections. This approach can be used to learn patterns at intersections or roundabouts from an onboard perspective to predict if the oncoming vehicles will exit the intersection or the roundabout or if they will cross in front of the ego-vehicle.

- Trajectory prediction results have shown that prediction performance grows with the U-net depth levels. Due to hardware limitations, higher depth levels could not be either trained or tested. More efficient representations will allow trying these deeper models. Hardware improvements will allow us to try these models.
- A hybrid model that joins both trajectory and maneuver prediction concepts in a single graphic representation could improve both predictions because trajectories are related to maneuvers and vice versa.

Appendix A

Confusion Matrix

This chapter collects the confusion matrix for all the experiments conducted in section 6.1.1. The chapter is divided in sections based on the name of the corresponding netowrk. Each confusion matrix is denoted by the name of the network, the training set $S = 1, 2, 3$ and the prediction time $t_p = 0, 10$.

A.1 SqueezeNet Confusion Matrix

Table A.1: Confusion Matrix, Squeezenet, $S = 1$, $t_p = 0$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	57161	1999	2578	0.926
left	7228	6422	1054	0.437
right	8809	1026	9428	0.489
Precision	0.781	0.680	0.722	0.763

Table A.2: Confusion Matrix, Squeezenet, $S = 2$, $t_p = 0$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	49692	1292	1213	0.952
left	7814	5917	733	0.409
right	7491	707	7939	0.492
Precision	0.765	0.747	0.803	0.768

Table A.3: Confusion Matrix, SqueezeNet, $S = 3$, $t_p = 0$

Test Fold		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**All**
	Acc	80.4	75.6	78.8	72.4	71.5	76.0	78.3	75.7	76.4	77.3	75.1	76.8
none	**Pre**	90.7	92.4	96.3	91.2	93.6	93.0	96.3	94.3	95.9	95.0	92.6	94.3
	Rec	83.8	77.0	77.5	72.7	73.5	77.8	77.7	74.2	75.3	77.7	74.0	77.2
left	**Pre**	39.0	43.3	45.7	43.8	52.2	27.3	42.8	49.3	36.7	42.3	61.4	40.7
	Rec	57.0	71.4	79.7	64.4	54.4	77.2	79.5	67.2	77.4	82.8	76.1	74.3
right	**Pre**	72.3	51.8	55.4	57.1	37.5	54.7	44.9	48.2	50.5	49.0	37.5	51.0
	Rec	78.0	71.1	86.1	77.8	74.8	66.3	82.8	88.3	82.4	69.9	83.1	76.1

Table A.4: Confusion Matrix Squeezenet, $S = 1$, $t_p = 10$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	58459	1672	2408	0.935
left	11558	7376	1381	0.363
right	12997	1640	11087	0.431
Precision	0.704	0.690	0.745	0.708

Table A.5: Confusion Matrix Squeezenet, $S = 2$, $t_p = 10$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	52902	1243	1168	0.956
left	10720	6708	1103	0.362
right	10419	954	8909	0.439
Precision	0.714	0.753	0.797	0.728

Table A.6: Confusion Matrix Squeezenet, $S = 3$, $t_p = 10$

Test Fold		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**All**
	Acc	71.7	71.9	78.9	72.1	69.0	74.9	78.8	72.8	76.2	77.5	74.1	75.6
none	**Pre**	89.3	92.5	94.6	92.7	93.4	93.0	95.8	92.9	94.9	94.4	91.5	93.8
	Rec	73.9	72.6	79.8	70.9	68.7	76.0	78.9	72.1	76.1	78.9	74.1	76.3
left	**Pre**	29.3	32.9	58.4	44.8	52.1	26.9	44.0	43.7	34.2	44.5	58.3	39.5
	Rec	62.9	84.3	65.1	64.1	66.6	71.8	75.7	65.5	76.0	73.3	71.4	71.8
right	**Pre**	59.6	62.7	46.1	54.3	35.7	50.9	43.5	45.4	52.0	46.1	37.6	48.4
	Rec	67.6	59.2	84.8	83.1	72.3	69.9	80.6	80.7	76.9	71.3	79.0	74.0

A.2 Googlenet Confusion Matrix

Table A.7: Confusion Matrix Googlenet, $S = 1$, $t_p = 0$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	57161	1999	2578	0.926
left	7228	6422	1054	0.437
right	8809	1026	9428	0.489
Precision	0.781	0.680	0.722	0.763

Table A.8: Confusion Matrix Googlenet, $S = 2$, $t_p = 0$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	48823	1254	1752	0.942
left	8766	5911	712	0.384
right	7408	751	7421	0.476
Precision	0.751	0.747	0.751	0.751

Table A.9: Confusion Matrix Googlenet, $S = 3$, $t_p = 0$

Test Fold		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**All**
	Acc	75.8	81.0	82.6	76.3	72.9	75.5	79.6	79.1	80.1	78.9	69.5	77.8
none	**Pre**	90.9	90.9	95.7	88.7	93.1	94.1	93.7	92.4	93.8	92.2	91.9	93.0
	Rec	80.7	86.2	82.8	81.0	74.2	76.3	81.2	79.9	82.1	81.7	66.7	79.8
left	**Pre**	29.0	54.3	59.2	52.1	48.2	28.5	50.2	51.4	38.9	45.0	52.8	43.0
	Rec	61.4	68.1	77.4	59.7	59.7	74.4	65.5	69.0	60.4	70.2	77.8	67.4
right	**Pre**	68.6	59.1	55.1	60.5	42.7	49.9	42.6	55.8	55.7	51.8	32.5	51.0
	Rec	63.5	63.8	85.5	72.6	76.5	71.1	80.2	81.4	78.9	67.2	77.8	73.2

Table A.10: Confusion Matrix Googlenet, $S = 1$, $t_p = 10$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	62543	3020	3168	0.910
left	8928	6404	1381	0.383
right	11543	1264	10327	0.446
Precision	0.753	0.599	0.694	0.730

Table A.11: Confusion Matrix Googlenet, $S = 2$, $t_p = 10$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	62253	3054	2910	0.913
left	8933	5231	2357	0.317
right	11828	2403	9609	0.403
Precision	0.750	0.489	0.646	0.710

Table A.12: Confusion Matrix Googlenet, $S = 3$, $t_p = 10$

Test Fold		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**All**
	Acc	73.1	75.5	78.3	73.9	74.8	75.2	78.8	73.1	75.6	77.4	71.4	76.0
none	**Pre**	89.0	89.6	95.4	91.2	91.2	92.2	94.6	91.6	94.3	92.1	88.2	92.5
	Rec	74.2	80.7	78.4	75.5	80.1	78.0	79.9	73.9	76.0	80.3	73.5	78.1
left	**Pre**	32.3	41.9	53.7	48.7	45.9	26.1	46.6	34.1	39.6	40.4	60.7	40.7
	Rec	53.7	62.2	66.8	59.6	50.7	59.0	65.2	68.3	58.4	69.3	58.9	61.9
right	**Pre**	58.6	51.4	46.8	54.3	45.6	48.3	41.3	55.7	44.4	51.4	32.4	47.5
	Rec	77.7	59.0	87.4	79.7	67.7	66.1	82.2	72.2	81.9	65.0	76.3	73.8

A.3 Alexnet Confusion Matrix

Table A.13: Confusion Matrix, Alexnet, $S = 1$, $t_p = 0$

	Output Class			
Target Class	***none***	***left***	***right***	**Recall**
none	55405	2236	2556	0.920
left	9207	5543	2348	0.324
right	8586	1668	8156	0.443
Precision	0.757	0.587	0.625	0.722

Table A.14: Confusion Matrix, Alexnet, $S = 2$, $t_p = 0$

	Output Class			
Target Class	***none***	***left***	***right***	**Recall**
none	46115	1004	1029	0.958
left	8047	5469	597	0.388
right	10835	1443	8259	0.402
Precision	0.709	0.691	0.836	0.723

Table A.15: Confusion Matrix, Alexnet, $S = 3$, $t_p = 0$

Test Fold		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**All**
	Acc	70.8	71.2	74.1	66.5	75.0	76.7	70.3	78.1	77.6	76.4	70.7	73.8
none	**Pre**	87.0	91.6	95.3	91.2	92.6	93.0	96.7	92.6	94.2	93.0	90.9	93.3
	Rec	73.8	72.3	72.9	65.8	81.7	79.2	68.2	79.5	79.3	78.3	70.1	74.9
left	**Pre**	27.9	33.8	44.2	38.5	34.6	25.8	38.8	44.6	35.1	40.2	56.6	36.4
	Rec	62.4	74.9	77.5	60.6	53.7	63.8	74.5	69.5	58.6	68.5	62.5	66.8
right	**Pre**	61.9	53.0	43.9	47.7	50.8	54.2	31.4	56.9	50.6	47.6	33.6	45.9
	Rec	63.1	62.5	78.0	73.9	58.4	68.4	84.5	77.0	78.1	70.1	90.0	72.4

Table A.16: Confusion Matrix, Alexnet, $S = 1$, $t_p = 10$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	62253	3054	2910	0.913
left	8933	5231	2357	0.317
right	11828	2403	9609	0.403
Precision	0.750	0.489	0.646	0.710

Table A.17: Confusion Matrix, Alexnet, $S = 2$, $t_p = 10$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	54132	1562	1693	0.943
left	11871	5632	1441	0.297
right	8038	1711	8046	0.452
Precision	0.731	0.632	0.720	0.720

Table A.18: Confusion Matrix Alexnet, $S = 3$, $t_p = 10$

Test Fold		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**All**
	Acc	69.6	66.5	76.3	66.2	70.9	74.7	75.9	70.1	73.7	73.4	69.8	73.1
none	**Pre**	80.9	87.8	94.5	90.5	90.2	92.6	93.8	97.0	94.1	93.3	91.8	91.8
	Rec	80.4	70.1	77.3	66.0	75.5	77.8	77.8	68.6	74.4	74.9	68.5	75.6
left	**Pre**	25.8	26.6	49.2	35.0	41.8	21.4	42.3	33.6	31.9	41.5	56.8	34.7
	Rec	48.5	55.5	63.6	57.7	53.3	64.1	55.9	69.4	60.7	60.3	64.9	58.8
right	**Pre**	61.0	46.0	43.8	49.9	40.0	54.7	35.8	44.7	44.8	38.7	32.3	43.6
	Rec	38.4	56.1	80.8	74.3	62.3	61.5	79.3	78.2	76.4	74.7	88.3	67.8

A.4 Resnet18 Confusion Matrix

Table A.19: Confusion Matrix Resnet18, $S = 1$, $t_p = 0$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	55405	2236	2556	0.920
left	9207	5543	2348	0.324
right	8586	1668	8156	0.443
Precision	0.757	0.587	0.625	0.722

Table A.20: Confusion Matrix Resnet18, $S = 2$, $t_p = 0$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	60472	3025	2917	0.911
left	1855	4281	244	0.671
right	2670	610	6724	0.672
Precision	0.930	0.541	0.680	0.863

Table A.21: Confusion Matrix Resnet18, $S = 3$, $t_p = 0$

Test Fold		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**All**
	Acc	83.0	85.9	86.5	81.5	84.6	84.3	88.5	82.9	87.0	85.1	82.4	85.5
none	**Pre**	86.5	89.2	91.4	87.7	90.7	90.0	92.1	88.5	91.4	88.4	87.9	90.0
	Rec	92.2	94.5	92.6	90.1	93.8	91.4	94.6	90.3	93.0	94.4	89.7	93.0
left	**Pre**	49.5	65.9	79.7	64.9	67.8	40.3	73.9	61.6	60.1	69.8	72.0	62.2
	Rec	52.1	61.8	57.2	49.8	46.9	56.6	55.4	48.6	58.9	49.2	58.3	54.3
right	**Pre**	85.6	80.2	64.1	69.9	64.7	73.4	66.4	65.8	74.1	67.2	55.5	70.7
	Rec	61.8	59.5	74.8	76.1	68.6	56.3	67.5	68.0	67.9	53.2	64.8	63.2

Table A.22: Confusion Matrix Resnet18, $S = 1$, $t_p = 10$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	77714	5520	6626	0.865
left	2612	4721	709	0.587
right	2688	447	7541	0.706
Precision	0.936	0.442	0.507	0.829

Table A.23: Confusion Matrix Resnet18, $S = 2$, $t_p = 10$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	68085	3490	3542	0.906
left	2939	4753	421	0.586
right	3017	662	7217	0.662
Precision	0.920	0.534	0.646	0.851

Table A.24: Confusion Matrix Resnet18, $S = 3$, $t_p = 10$

Test Fold		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**All**
	Acc	81.0	82.1	85.2	79.2	84.5	84.7	86.9	78.7	86.4	83.3	80.6	84.2
none	**Pre**	86.7	85.2	91.7	85.6	89.8	88.9	91.3	86.4	90.3	88.1	85.3	89.0
	Rec	89.1	94.2	91.0	88.3	92.8	93.0	93.7	87.5	93.7	92.7	90.7	92.5
left	**Pre**	44.5	59.3	73.7	56.9	73.3	42.1	72.1	47.1	57.3	60.2	74.2	59.0
	Rec	58.5	43.7	59.4	48.8	59.6	50.6	48.6	44.8	49.7	40.3	53.5	49.7
right	**Pre**	80.4	74.3	59.5	70.2	65.8	76.6	58.7	59.5	75.2	60.5	48.9	67.7
	Rec	60.7	48.9	72.0	70.1	64.4	51.7	64.6	57.7	64.1	56.1	48.0	59.6

A.5 Resnet50 Confusion Matrix

Table A.25: Confusion Matrix Resnet50, $S = 1$, $t_p = 0$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	68654	4420	5052	0.879
left	2113	4490	527	0.630
right	2431	537	7481	0.716
Precision	0.938	0.475	0.573	0.842

Table A.26: Confusion Matrix Resnet50, $S = 2$, $t_p = 0$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	60794	3174	2798	0.911
left	1917	4212	167	0.669
right	2286	530	6920	0.711
Precision	0.935	0.532	0.700	0.869

Table A.27: Confusion Matrix Resnet50, $S = 2$, $t_p = 0$

Test Fold		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**All**
	Acc	86.4	85.8	85.7	82.9	86.3	86.2	88.9	84.5	88.1	87.5	84.0	86.9
none	**Pre**	87.2	90.2	92.6	87.1	89.8	90.9	93.3	90.5	94.0	90.1	87.9	91.1
	Rec	96.1	92.3	90.4	92.1	95.1	92.8	93.9	89.9	91.9	95.5	91.7	93.4
left	**Pre**	72.7	65.1	64.2	74.9	73.5	42.3	70.1	64.6	64.4	72.5	73.3	65.3
	Rec	41.4	62.4	62.7	51.7	59.0	54.9	62.4	51.1	61.5	59.2	58.6	57.9
right	**Pre**	87.1	76.9	66.1	71.3	74.5	79.7	68.3	67.3	70.3	75.9	65.8	74.0
	Rec	70.0	70.0	76.7	74.6	64.4	61.7	72.2	78.8	80.5	57.7	65.7	68.8

Table A.28: Confusion Matrix Resnet50, $S = 1$, $t_p = 10$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	77205	5190	6491	0.869
left	2834	4806	686	0.577
right	2975	692	7699	0.677
Precision	0.930	0.450	0.518	0.826

Table A.29: Confusion Matrix Resnet50, $S = 2$, $t_p = 10$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	68955	3649	3614	0.905
left	2382	4675	340	0.632
right	2704	581	7226	0.687
Precision	0.931	0.525	0.646	0.859

Table A.30: Confusion Matrix Resnet50, $S = 3$, $t_p = 10$

Test Fold		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**All**
	Acc	80.5	81.9	86.8	81.4	85.7	84.2	86.5	80.5	86.3	84.3	80.3	84.4
none	**Pre**	86.0	84.1	91.8	88.0	89.5	89.1	91.6	85.3	91.2	89.2	84.8	89.3
	Rec	89.4	94.2	92.9	89.8	95.9	92.1	92.6	91.5	92.8	92.6	90.8	92.3
left	**Pre**	41.8	61.8	76.1	62.1	72.0	41.7	69.2	51.6	55.0	61.6	73.5	59.0
	Rec	50.5	28.4	64.3	54.3	53.3	50.3	51.9	35.1	55.6	52.9	50.4	51.4
right	**Pre**	79.7	75.2	65.0	70.0	72.7	72.7	57.3	65.8	73.4	64.4	50.2	68.2
	Rec	60.2	59.0	68.6	71.8	62.2	53.5	67.0	56.0	65.7	55.6	49.5	60.7

A.6 Resnet101 Confusion Matrix

Table A.31: Confusion Matrix Resnet101, $S = 1$, $t_p = 0$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	66858	3450	4144	0.898
left	3122	5337	431	0.600
right	3218	660	8485	0.686
Precision	0.913	0.565	0.650	0.843

Table A.32: Confusion Matrix Resnet101, $S = 2$, $t_p = 0$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	60523	2499	3019	0.916
left	2816	4998	340	0.613
right	1658	419	6526	0.759
Precision	0.931	0.631	0.660	0.870

Table A.33: Confusion Matrix Resnet101, $S = 3$, $t_p = 0$

Test Fold		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**All**
	Acc	78.7	81.3	83.7	80.9	80.1	83.8	85.0	81.6	85.9	81.5	73.1	82.7
none	**Pre**	87.4	90.0	92.8	91.5	93.0	91.8	94.5	92.5	90.6	91.2	88.2	91.7
	Rec	86.8	88.6	89.1	86.5	87.5	90.1	88.4	85.8	93.9	87.7	77.2	88.6
left	**Pre**	41.1	62.2	66.8	61.7	48.2	32.1	47.5	45.9	49.7	42.1	63.1	46.7
	Rec	32.4	54.4	46.3	52.2	65.5	58.9	68.5	69.0	53.4	63.6	55.7	57.9
right	**Pre**	61.2	54.5	54.2	63.0	54.4	72.8	62.1	66.1	75.4	65.2	30.0	62.1
	Rec	68.8	63.9	82.6	83.6	55.5	57.6	70.8	68.7	57.2	55.1	70.0	64.1

Table A.34: Confusion Matrix Resnet101, $S = 1, t_p = 10$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	75830	5179	4902	0.883
left	2187	4745	665	0.625
right	4997	764	9309	0.618
Precision	0.913	0.444	0.626	0.828

Table A.35: Confusion Matrix Resnet101, $S = 2, t_p = 10$

	Output Class			
Target Class	*none*	*left*	*right*	**Recall**
none	68571	3732	3322	0.907
left	2326	4451	317	0.627
right	3144	722	7541	0.661
Precision	0.926	0.500	0.675	0.856

Table A.36: Confusion Matrix Resnet101, $S = 3, t_p = 10$

Test Fold		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**All**
	Acc	84.5	85.6	85.9	82.8	81.0	84.1	78.0	80.6	80.7	84.7	84.1	84.1
none	**Pre**	90.1	93.5	92.5	88.0	84.4	91.0	89.0	88.4	87.4	87.8	88.7	90.1
	Rec	91.5	89.4	90.9	89.9	92.8	89.8	84.3	87.7	88.8	94.4	94.9	90.9
left	**Pre**	40.2	63.6	56.1	45.7	60.5	63.0	59.6	60.0	74.7	71.0	78.0	58.2
	Rec	59.9	62.8	56.5	56.0	34.9	64.7	47.7	35.3	54.7	41.2	49.9	53.5
right	**Pre**	76.0	53.0	67.2	83.6	68.6	62.8	58.0	57.2	46.4	67.1	62.7	65.0
	Rec	55.4	74.7	73.3	68.2	55.3	66.2	79.2	75.2	61.9	56.8	58.7	65.0

Appendix B

Publications Derived from this PhD Dissertation

B.1 Journal Publications

2020 ***Submitted - Vehicle Trajectory and Maneuver Prediction in Highway Scenarios for Highly Automated Vehicles**, *Izquierdo, Rubén and Quintanar, Álvaro and Parra, Ignacio and Ferández-Llorca, David and Sotelo, Miguel Ángel*, IEEE Transactions on Intelligent Transportation Systems.

2020 **Simple Baseline for Vehicle Pose Estimation: Experimental Validation**, *H. Corrales, A. Hernández, R. Izquierdo, N. Hernández, I. Parra, D. F. Llorca*, IEEE Access.

2020 **Fail-aware LIDAR-based odometry for autonomous vehicles**, *I. G. Daza, M. Rentero, C. Salinas, R. Izquierdo, N. Hernández, A. Ballardini, D. F. Llorca*, Sensors.

2020 **License Plate Corners Localization Using CNN-Based Regression**, *D. F. Llorca, H. Corrales, I. Parra, M. Rentero, R. Izquierdo, A. Hernández, I. García-Daza*, Lecture Notes in Computer Science.

2018 **High-Level Interpretation of Urban Road Maps Fusing Deep Learning-Based Pixelwise Scene Segmentation and Digital Navigation Maps**, *C. Fernández, J. Muñoz-Bulnes, D. F. Llorca, I. Parra, I. García-Daza, R. Izquierdo, M. A. Sotelo*, Journal of Advanced Transportation.

2017 ***The Experience of DRIVERTIVE-DRIVERless cooperaTIve VEhicle-Team in the 2016 GCDC**, *Alonso, Ignacio Parra and Izquierdo, Rubén and Alonso, Javier and García-Morcillo, Álvaro and Fernández-Llorca, David and Sotelo, Miguel Ángel*, IEEE Transactions on Intelligent Transportation Systems.

B.2 Conference Publications

2020 ***Vehicle Trajectory Prediction in Crowded Highway Scenarios Using Bird Eye View Representations and CNNs**, *Izquierdo, Rubén and Quintanar, Álvaro and Parra, Ignacio and Fernández-Llorca, David and Sotelo, Miguel Ángel*, 2020 IEEE Intelligent Transportation Systems Conference (ITSC).

2020 **Two-Stream Networks for Lane-Change Prediction of Surrounding Vehicles**, *D. F. Llorca, M. Biparva, R. Izquierdo, J. K. Tsotsos*, 2020 IEEE Intelligent Transportation Systems Conference (ITSC).

2020 ***The PREVENTION Challenge: How Good Are Humans PredictingLane Changes?**, *A. Quintanar, R. Izquierdo, I. Parra, D. F. Llorca, M. A. Sotelo*, 2020 IEEE Intelligent Vehicle Symposium 2020 (IV).

2019 ***Experimental validation of lane-change intention prediction methodologies based on CNN and LSTM**, *Izquierdo, Rubén and Quintanar, Álvaro and Parra, Ignacio and Fernández-Llorca, David and Sotelo, Miguel Ángel*, 2019 IEEE Intelligent Transportation Systems Conference (ITSC).

2019 ***The PREVENTION dataset: a novel benchmark for PREdiction of VEhicles iNTentIONs**, *Izquierdo, Rubén and Quintanar, Álvaro and Parra, Ignacio and Fernández-Llorca, David and Sotelo, Miguel Ángel*, 2019 IEEE Intelligent Transportation Systems Conference (ITSC).

2018 ***Multi-Radar Self-Calibration Method using High-Definition Digital Maps for Autonomous Driving**, *Izquierdo, Rubén and Parra, Ignacio and Fernández-Llorca, David and Sotelo, Miguel Ángel*, 2018 IEEE 21st International Conference on Intelligent Transportation Systems (ITSC).

2018 ***Semi-Automatic High-Accuracy Labelling Tool for Multi-Modal Long-Range Sensor Dataset**, *Izquierdo, Rubén and Parra, Ignacio and Salinas, C and Fernández-Llorca, David and Sotelo, Miguel Ángel*, 2018 IEEE Intelligent Vehicles Symposium (IV).

2017 ***Vehicle Trajectory and Lane Change Prediction Using ANN and SVM Classifiers**, *Izquierdo, Rubén and Parra, Ignacio and Muñoz-Bulnes, Jesús and Fernández-Llorca, David and Sotelo, Miguel Ángel*, 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC).

2017 **Analysis of ITS-G5A V2X communications performance in autonomous cooperative driving experiments**, *I. Parra, A. García-Morcillo, R. Izquierdo, J. Alonso, D. F. Llorca, M. A. Sotelo*, 2017 IEEE Intelligent Vehicles Symposium (IV).

2016 **Two-camera based accurate vehicle speed measurement using average speed at a fixed point**, *D. F. Llorca, C. Salinas, M. Jiménez, I. Parra, A. G. Morcillo, R. Izquierdo, J. Lorenzo, M. A. Sotelo*, 2016 IEEE International Conference on Intelligent Transportation Systems (ITSC).

2016 **Fusing directional passive UHF RFID and stereo vision for tag association in outdoor scenarios**, *D. F. Llorca, R. Quintero, I. Parra, M. Jiménez, C. Fernández, R. Izquierdo, M. A. Sotelo*, 2016 IEEE International Conference on Intelligent Transportation Systems (ITSC).

2016 **Comparison between UHF RFID and BLE for stereo-based tag association in outdoor scenarios**, *D. F. Llorca, R. Quintero, I. Parra, C. Fernández, R. Izquierdo, M. A. Sotelo*, 6th International Conference on IT Convergence and Security (ICITCS).

2016 **Drivertive Team in the Grand Cooperative Driving Challenge 2016**, *J. Alonso, I. Parra, R. Izquierdo, A. García, M. A. Sotelo, C. Fernández, R. Quintero, C. Salinas, J. Lorenzo, D. F. Llorca*, 1st Symposium SEGVAUTO-TRIES-CM: Technologies for a Safe, Accessible and Sustainable Mobility.

* Publications directly related to this thesis.

Bibliography

[1] “Annual energy outlook 2019,” U.S. Energy Information Administration, Tech. Rep., 2019. [Online]. Available: https://www.eia.gov/outlooks/aeo/pdf/aeo2019.pdf

[2] “Global status report on road safety 2018,” World Health Organization, Tech. Rep., 2018. [Online]. Available: http://www.who.int/violence_injury_prevention/road_safety_status/2018/en/

[3] “International Energy Agency | statistics,” 2014.

[4] “Greenhouse gas emissions,” European Environment Agency, Tech. Rep., 2019. [Online]. Available: https://www.eea.europa.eu/airs/2018/resource-efficiency-and-low-carbon-economy/greenhouse-gas-emission#tab-based-on-data

[5] S. C. Davis, S. W. Diegel, R. G. Boundy, *et al.*, “Transportation energy data book,” Oak Ridge National Laboratory, Tech. Rep., 2009.

[6] “Traffic scorecard ranking,” Inrix Global, Tech. Rep., 2018. [Online]. Available: http://inrix.com/scorecard/

[7] P. Christidis, J. N. I. Rivas, *et al.*, “Measuring road congestion,” *Institute for Prospective and Technological Studies, Joint Research Centre, Brussels*, 2012.

[8] J. Halkias and J. Colyar, https://www.fhwa.dot.gov/publications/research/operations/06137/06137.pdf, December 2006, nGSIM - Interstate 80 Freeway Dataset [Online; accessed Jan. 9 2019].

[9] J. Colyar and J. Halkias, “Ngsim - us highway 101 dataset,” Jan 2007, https://www.fhwa.dot.gov/publications/research/operations/07030/07030.pdf [Online; accessed Jan. 9 2019].

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[12] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017. [Online]. Available: http://dx.doi.org/10.1177/0278364916679498

[13] U. of Peking, "University of peking database (pku)," http://poss.pku.edu.cn/download/.

[14] A. Rangesh, K. Yuen, R. K. Satzoda, R. N. Rajaram, P. Gunaratne, and M. M. Trivedi, "A multimodal, full-surround vehicular testbed for naturalistic studies and benchmarking: Design, calibration and deployment," *arXiv preprint arXiv:1709.07502*, 2017.

[15] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang, "The apolloscape dataset for autonomous driving," *arXiv preprint arXiv:1803.06184*, 2018.

[16] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," 2018.

[17] V. Ramanishka, Y.-T. Chen, T. Misu, and K. Saenko, "Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[18] A. Patil, S. Malla, H. Gang, and Y.-T. Chen, "The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes," in *International Conference on Robotics and Automation*, 2019.

[19] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *2018 IEEE 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[20] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[21] "Waymo open dataset: An autonomous driving dataset," 2019.

[22] J. Wiest, M. Höffken, U. Kresel, and K. Dietmayer, "Probabilistic trajectory prediction with gaussian mixture models," in *2012 IEEE Intelligent Vehicles Symposium*, 2012.

[23] E. Yurtsever, Y. Liu, J. Lambert, C. Miyajima, E. Takeuchi, K. Takeda, and J. H. L. Hansen, "Risky action recognition in lane change video clips using deep spatiotemporal networks with segmentation mask transfer," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.

[24] Z. Wei, C. Wang, P. Hao, and M. J. Barth, "Vision-based lane-changing behavior detection using deep residual neural network," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.

[25] C. Hermes, C. Wohler, K. Schenk, and F. Kummert, "Long-term vehicle motion prediction," in *2009 IEEE Intelligent Vehicles Symposium*, June 2009, pp. 652–657.

[26] S. Ammoun and F. Nashashibi, "Real time trajectory prediction for collision risk estimation between vehicles," in *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*, 2009.

[27] R. S. Tomar, S. Verma, and G. S. Tomar, "Prediction of lane change trajectories through neural network," in *2010 International Conference on Computational Intelligence and Communication Networks*, Nov 2010, pp. 249–253.

[28] A. Houenou, P. Bonnifait, V. Cherfaoui, and W. Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 4363–4369.

[29] W. Yao, H. Zhao, P. Bonnifait, and H. Zha, "Lane change trajectory prediction by using recorded human driving data," in *2013*

IEEE Intelligent Vehicles Symposium (IV), June 2013, pp. 430–436.

[30] S. Yoon and D. Kum, "The multilayer perceptron approach to lateral motion prediction of surrounding vehicles for autonomous vehicles," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, June 2016, pp. 1307–1312.

[31] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

[32] F. Altche and A. de La Fortelle, "An lstm network for highway trajectory prediction," in *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

[33] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms," in *IEEE Intelligent Vehicle Symposium (IVS)*, 2018, pp. 1179–1184.

[34] Y. Hu, W. Zhan, and M. Tomizuka, "A framework for probabilistic generic traffic scene prediction," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2790–2796.

[35] P. Schörner, L. Töttel, J. Doll, and J. M. Zöllner, "Predictive trajectory planning in situations with hidden road users using partially observable markov decision processes," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[36] A. Benterki, V. Judalet, M. Choubeila, and M. Boukhnifer, "Long-term prediction of vehicle trajectory using recurrent neural networks," in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, 2019.

[37] D. Roy, T. Ishizaka, C. K. Mohan, and A. Fukuda, "Vehicle trajectory prediction at intersections using interaction based generative adversarial networks," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.

[38] J. Schlechtriemen, A. Wedel, J. Hillenbrand, G. Breuel, and K. D. Kuhnert, "A lane change detection approach using feature ranking with maximized predictive power," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, June 2014, pp. 108–114.

[39] D. Kasper, G. Weidl, T. Dang, G. Breuel, A. Tamke, A. Wedel, and W. Rosenstiel, "Object-oriented bayesian networks for detection of lane change maneuvers," *IEEE Intelligent Transportation Systems Magazine*, vol. 4, no. 3, 2012.

[40] R. Graf, H. Deusch, M. Fritzsche, and K. Dietmayer, "A learning concept for behavior prediction in traffic situations," in *IEEE Intelligent Vehicle Symposium (IVS)*, 2013, pp. 672–677.

[41] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier, "Learning-based approach for online lane change intention prediction," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 797–802.

[42] J. Schlechtriemen, F. Wirthmueller, A. Wedel, G. Breuel, and K.-D. Kuhnert, "When will it change the lane? a probabilistic regression approach for rarely occurring events," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 1373–1379.

[43] S. Yoon and D. Kum, "The multilayer perceptron approach to lateral motion prediction of surrounding vehicles for autonomous vehicles," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, June 2016, pp. 1307–1312.

[44] M. Bahram, C. Hubmann, A. Lawitzky, M. Aeberhard, and D. Wollherr, "A combined model- and learning-based framework for interaction-aware maneuver prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1538–1550, June 2016.

[45] V. Leonhardt and G. Wanielik, "Neural network for lane change prediction assessing driving situation, driver behavior and vehicle movement," in *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*. IEEE, 2017, pp. 1–6.

[46] H. Zhao, C. Wang, Y. Lin, F. Guillemard, S. Geronimi, and F. Aioun, "On-road vehicle trajectory collection and scene-based lane change analysis: Part i," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 1, pp. 192–205, Jan 2017, http://poss.pku.edu.cn/download [Online; accessed Jan. 9 2019].

[47] W. Yao, Q. Zeng, Y. Lin, D. Xu, H. Zhao, F. Guillemard, S. Geronimi, and F. Aioun, "On-road vehicle trajectory collection and

scene-based lane change analysis: Part ii," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 1, pp. 206–220, Jan 2017.

[48] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, 2018, pp. 1468–1476.

[49] S. Patel, B. Griffin, K. Kusano, and J. J. Corso, "Predicting future lane changes of other highway vehicles using rnn-based deep models," 2018.

[50] J. Li, B. Dai, X. Li, X. Xu, and D. Liu, "A dynamic bayesian network for vehicle maneuver prediction in highway driving scenarios: Framework and verification," *Electronics*, vol. 8, no. 40, 2019.

[51] M. Kruger, A. S. Novo, T. Nattermann, and T. Bertram, "Probabilistic lane change prediction using gaussian process neural networks," in *IEEE 22th International Conference on Intelligent Transportation Systems (ITSC)*, 2019, pp. 3651–3656.

[52] J. Li, C. Lu, Y. Xu, Z. Zhang, J. Gong, and H. Di, "Manifold learning for lane-changing behavior recognition in urban traffic," in *IEEE 22th International Conference on Intelligent Transportation Systems (ITSC)*, 2019, pp. 3663–3668.

[53] D. Lee, Y. P. Kwon, S. McMains, and J. K. Hedrick, "Convolution neural network-based lane change intention prediction of surrounding vehicles for acc," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017, pp. 1–6.

[54] G. L. Smith, S. F. Schmidt, and L. A. McGee, *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle.* National Aeronautics and Space Administration, 1962.

[55] R. Pepy, A. Lambert, and H. Mounier, "Path planning using a dynamic vehicle model," in *2006 2nd International Conference on Information Communication Technologies*, vol. 1, April 2006, pp. 781–786.

[56] I. P. Alonso, R. I. Gonzalo, J. Alonso, Á. García-Morcillo, D. Fernández-Llorca, and M. Á. Sotelo, "The experience of

drivertive-driverless cooperative vehicle-team in the 2016 gcdc," *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[57] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the nelder–mead simplex method in low dimensions," *SIAM Journal on optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[58] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron," https://github.com/facebookresearch/detectron, 2018.

[59] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[61] H. W. Kuhn, "Kuhn hw. the hungarian method for the assignment problem. naval research logistic quaterly 1955; 2: 83-97," *Naval Research Logistic Quaterly*, vol. 2, pp. 83–97, 1955.

[62] C. Fernández, R. Izquierdo, D. F. Llorca, and M. A. Sotelo, "Road curb and lanes detection for autonomous driving on urban scenarios," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct 2014, pp. 1964–1969.

[63] R. Izquierdo, I. Parra, C. Salinas, D. FernÃ¡ndez-Llorca, and M. A. Sotelo, "Semi-automatic high-accuracy labelling tool for multi-modal long-range sensor dataset," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, June 2018, pp. 1786–1791.

[64] R. Izquierdo, I. Parra, D. Fernández-Llorca, and M. A. Sotelo, "Multi-radar self-calibration method using high-definition digital maps for autonomous driving," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2018, pp. 2197–2202.

[65] S. De Craen, D. A. Twisk, M. P. Hagenzieker, H. Elffers, and K. A. Brookhuis, "Do young novice drivers overestimate their driving skills more than experienced drivers? Different methods lead to different conclusions," *Accident Analysis and Prevention*, vol. 43, no. 5, pp. 1660–1665, sep 2011.

[66] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[67] “Brave - bridging gaps for the adoption of automated vehicles,” European Union’s Horizon 2020, Tech. Rep., N 723021. [Online]. Available: https://www.brave-project.eu/

[68] “Utac-ceram.” [Online]. Available: https://www.utacceram.com/testing-expertise/safety/active