UNIVERSITÀ DI PARMA

UNIVERSITA' DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN

"Matematica"

CICLO XXXV

Advanced Driver Assistance Systems for high-performance Embedded Automotive Platforms

Coordinatore:

Prof. Alessandra Lunardi

Tutore:

Prof. Marko Bertogna and Eng. Paolo Burgio

Dottorando: Isa Moazen

Anni Accademici 2019/2020 – 2021/2022

There is a high demand of progressing in the world. Anyway, we should remember that the most important properties of us are the people who we love and the friends who help us grow. I would like to dedicate this thesis to my father who I loved but he did not have a chance to stay in this world to see me as a PhD. I would extend my thankfulness to my wife who tried to support me in the most difficult conditions of completing my thesis. I would like to also thank my kind supervisor, Paolo Burgio, who tried to stay beside me in the complicated moments and did not stop believing in me. Finally, thanks to Prof. Marko Bertogna for his right recommendations in each step, and all my classmates and professors who helped me to progress in this PhD as well as this part of life.

Isa Moazen, 01/2023

Abstract

Advanced driver-assistance systems (ADASs) have become a salient feature for safety in modern vehicles. They are also a key underlying technology in emerging autonomous vehicles. State-of-the-art ADASs are primarily vision based, but light detection and ranging (lidar), radio detection and ranging (radar), and other advanced-sensing technologies are also becoming popular. In the first chapter, we present a survey of different hardware and software ADAS technologies and their capabilities and limitations. In the first chapter, we discuss approaches used for vision-based recognition and sensor fusion in ADAS solutions. We also highlight challenges for the next generation of ADASs. Then in the second chapter, we discuss the high-performance embedded platforms using in automotive domain. Since there is a tight relationship between trust of the costumers and comfort in autonomous vehicles with the higher autonomy levels, we focused on the most important issue of the comfort, motion sickness, that impacts on many people. The outcome of our research work in the thesis was two novel methods to mitigate the motion sickness that we discuss in the third and fourth chapters. To extend our work, we decided to use the machine learning techniques for motion prediction. The motion prediction techniques had conducted us to use the traffic rules for having the outperformance in the intersections. Therefore, we developed a state-of-the-art motion prediction system that works in intersections that will be described in the fifth chapter.

In the third chapter a Motion Sickness mitigation system is introduced. Current full- and semi- Autonomous car prototypes increasingly feature complex algorithms for lateral and longitudinal control of the vehicle. Unfortunately, in some

cases, they might cause fussy and unwanted effects on the human body, such as motion sickness, ultimately harnessing passengers' comfort, and driving experience. Motion sickness is due to conflict between visual and vestibular inputs, and in the worst case might causes loss of control over one's movements, and reduced ability to anticipate the direction of movement. In the chapter two, we focus on the five main physical characteristics that affect motion sickness, including them in the function cost, to provide quality passengers' experience to vehicle passengers. We implemented our approach in a state-of-the-art Model Predictive Controller, to be used in a real Autonomous Vehicle. Preliminary tests using the Unreal Engine simulator have already shown that our approach is viable and effective, and we implemented and evaluated using Motion Sickness Dose Value and Illness Rating and then tested it in an embedded platform.

We have also developed another novel alerting system to minimize the motion sickness describing in the fourth chapter. Current intelligent car prototypes increasingly move to become autonomous where no driver is required. If an automated vehicle has rearward and forward-facing seats and none of the passengers pay attention to the road, they increasingly experience the motion sickness because of the inability of passengers to anticipate the future motion trajectory. In the chapter three, we focus on anticipatory audio and video cues using pleasant sounds and a Human Machine Interface to display and inform the passengers about the upcoming trajectories that may lead to make the passengers sick. To be able to anticipate the next moves, we require an evaluation system of the next 1 kilometer of the road using the map. The road is investigated based on the amount of the turns and the maximum speed allowed that lead to lateral accelerations that is high enough based on Motion Sickness Dose Value to make the passengers sick. The system alerts the passengers through a Human Machine Interface to focus on the road for prevention of the Motion Sickness. We evaluate our method by using Motion Sickness Dose Value. Based on this work, we can prevent the sickness due to lateral accelerations by making the passengers to focus on the road and decrease the vestibular conflict.

Finally, to extend our works into the machine learning techniques, in the fifth chapter, we started researching on motion prediction area and we developed a state-of-the-art motion prediction model. As declared, one of the motion sickness sources is ability to anticipate the direction of movement. Therefore, having a trustable prediction on the next trajectories, can even help decreasing the motion sickness and increasing the comfort. In the other hand, autonomous driving motion forecasting is essential to have a correct and reliable planning. The influence of the road agents on each other makes it even more challenging. However, most prior works have not considered these interactions and planning against fixed predictions would reduce the ability to represent the future interaction possibilities between different agents. In this chapter of the thesis, we propose a model that predicts the agents' behaviour in a jointly manner. We take advantage of using masking strategy as the query to our model. Our model architecture uses a unified Transformer architecture by employing attention across the road elements, agent interactions and traffic rules in intersections. We evaluate our approach on autonomous driving datasets for behavior prediction and test it on python. Our work demonstrates that motion forecasting by a model with a masking strategy and having attentions and traffic rules can lead us to a state-of-the-art model.

For the last three chapters mentioned above, I succeeded to publish the related publication as bellow:

1. Moazen, I., & Burgio, P. (2021). A Full-Featured, Enhanced Cost Function to Mitigate Motion Sickness in Semi-and Fully-autonomous Vehicles. In VEHITS (pp. 497-504).
2. Moazen, I., Burgio, P., & Castellano, A. (2022, August). Motion Sickness Minimization Alerting System Using The Next Curvature Topology. In 2022 IEEE International Conference on Mechatronics and Automation (ICMA) (pp. 635-640). IEEE.
3. Submitted: The Advantage of Using Traffic Rules for Motion Prediction in Intersections (TRMPI), In 2023 IEEE International Conference on Mechatronics and Automation (ICMA)

Contents

List of Figures

Chapter 1

Introduction to Advanced Driver Assistance Systems

1.1 Introduction

In modern vehicles advanced driver-assistance systems (ADASs) have become a crucial feature for safety. Developing them aims toward the technologies for autonomous vehicles. State-of-the-art ADASs are primarily vision based, but light detection and ranging (lidar), radio detection and ranging (radar), and other advanced-sensing technologies are also becoming popular. In this section, we try to categorize different hardware and software ADAS technologies and their capabilities and limitations. Finally, we introduce our embedded platform as a high-performance embedded platform that we developed three different ADASs on it.

1.1 How the safety impacts ADAS

Safety is always a crucial concern in automotive systems the early days of on-road vehicles. Many different research and development have been completed to address this issue by developing various safety systems to protect occupants within a vehicle as well as prevent injuries to people outside the vehicle. The safety systems can be categorized in passive (or reactive) and active (or proactive). Passive safety systems are the systems to protect the passengers of the vehicle from injuries after a crash. Some of them can be seat belts, air bags, and padded dashboards. These systems due to a consistent consumer demand for safer vehicles, are under continuous development. Anyway, to make the vehicles even safer, it is required to augmented them by active safety systems that tries to prevent a crash from

happening altogether. Active systems are one of the main areas of interest and have seen major growth in today's vehicles. Examples of such systems include lane keeping, automatic braking, and adaptive cruise control. These systems are commonly known as ADASs and are becoming increasingly popular as a way for automotive manufacturers to differentiate their offerings while promoting consumer safety.

Recent studies from the World Health Organization indicate that 1.25 million deaths occur every year due to road traffic accidents [1]. Moreover, such accidents in recent years have an annual global cost of US$518 billion, which takes away approximately 1–2% of gross domestic product from all of the countries in the world [2]. These high fatality rates, monetary losses, and increasing customer demand for intelligent safety systems are some of the key reasons for OEMs to develop ADASs. Moreover, with the increasing number of electronic control units and integration of various types of sensors, there are now sufficient computing capabilities in vehicles to support ADAS deployments. The different types of sensors, such as cameras, lidar, radar, and ultrasonic sensors, enable a variety of different ADAS solutions. Among them, the vision-based ADAS, which primarily uses cameras as vision sensors, is popular in most modern-day vehicles. Figure 1 shows some of the state-of-the art ADAS features and the sensors used to implement them.

Modern ADASs are also key technologies to realize autonomous vehicles [3]. But several challenges with the design, implementation, and operation of ADASs remain to be overcome. Some of these challenges include minimizing energy consumption, reducing response latency, adapting to changing weather conditions, and security. In this chapter, we describe the different ADASs along with their required sensors.

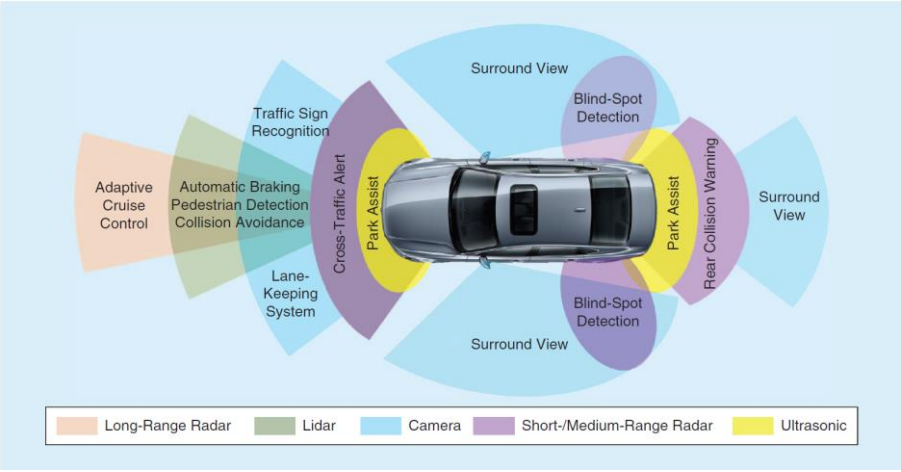

Fig. 1. The sensor ranges to fulfill the ADASs.

1.2 ADAS categories

The ADASs can be categorized based on the next groups:

1.2.1 Vision sensors

Cameras are the most commonly used vision sensors in vehicles. Vision-based ADAS uses one or more cameras to capture images and an embedded system to detect, analyze, and track different objects in them. In high-end ADAS, cameras are used to monitor both the inside and outside of the vehicle as shown in figure 2. Camera integration in modern vehicles is becoming more common because of its low cost and easy installation. Laws such as [4] (that mandate all vehicles manufactured from 1 May 2018 onward use vision-based ADAS) will further aid in camera integration. Cameras capture information such as color, contrast, and texture, which gives them a unique advantage over other sensors. Three types of cameras as shown in Figure 3 are often used in vision-based ADAS: 1) monocular, 2) stereo, and 3) IR cameras.

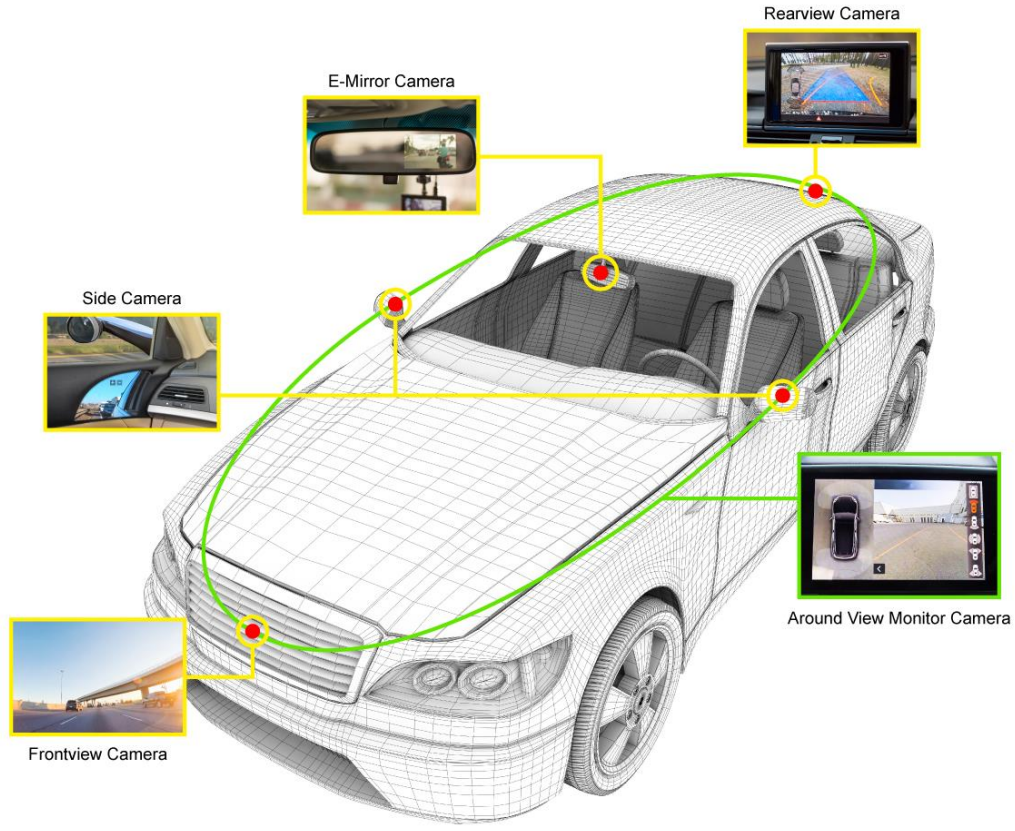

Fig. 2. The different vision sensors used in an intelligent vehicle.

1.2.2 Monocular cameras

These camera systems have only one lens. As these systems have only one image output at any point of time, they have low image-processing requirements compared to those of other camera types. These cameras can be used for multiple applications, such as the detection of obstacles, pedestrians, lanes, and traffic signs [5]. They can also be used for monitoring the driver inside a vehicle, e.g., for face- and eye-detection and head-pose analysis [6]. But monocular camera images lack depth information and are, therefore, not reliable sensors for distance estimation. Some techniques [5] allow approximating distance by identifying key features in the captured image frame and tracking their position when the camera is in motion.

Monocular vision systems are starting to emerge, but they are usually focusing only on one aspect of the problem e.g. lane departure warning. It turns out that in many situations providing warning based on one modality may be too limited. For example, lane departure system would gain a lot from insertion of information about vehicles on the road (blocking the view on the lanes). Furthermore, higher level of information about lanes can be of aid for example unstable driving within a lane (indicated by lateral velocity) may be an important indication of intelligent systems.

Range to vehicles and range-rate are two important values required for any vision-based system. As the data is collected from a single camera range must be estimated by using perspective. There are two cues which can be used: size of the vehicle in the image and position of the bottom of the vehicle in the image. Since the width of a vehicle of unknown type (car, van, truck etc) can vary anywhere between 1.5m and 3m a range estimate based on width will have only about 30% accuracy.

A much better estimate can be achieved using the road geometry and the point of contact of the vehicle with the road. We assume a planar road surface and a camera mounted so that the optical axis is parallel to the road surface. A point on the road at a distance Z in front of the camera will project to the image at a height y, where y is given by the equation:

$$y = \frac{fH}{Z} \qquad (1)$$

where H is the camera height, and f is the focal length of the camera (both given in meters).

1.2.3 Stereo cameras

These systems consist of two or more lenses, each with image sensors, separated by a certain distance (known as stereo base). Stereo cameras are useful in extracting three-dimensional (3-D) information from two or more two-dimensional images by matching stereo pairs (images from left and right sensors) and using a disparity map to estimate the relative depth of a scene. These cameras can be used for a variety of applications, such as traffic sign recognition, lane, pedestrian, and obstacle detection as well as distance estimation, with much greater accuracy compared to monocular cameras.

Stereo systems can be relied upon for accurate distance (depth) estimation over short distances, up to 30 m. In most production vehicles with stereo cameras, the cameras are located inside the vehicle, behind the rear-view mirror, angled slightly downward, and facing the road.

1.2.4 IR cameras

There are two main types of IR cameras. Active IR cameras use a near-IR light source (with wavelengths from 750 to 1,400 nm) built in the vehicle to illuminate the scene (which cannot be seen by the human eye) and a standard digital camera sensor to capture the reflected light. Passive IR cameras use an IR sensor, where every pixel on the IR sensor can be considered as a temperature sensor that can capture the thermal radiation emitted by any material. Unlike active IR cameras, passive IR cameras do not require any special illumination of the scene. Still, popular night-vision solutions mainly use active IR cameras to assist the driver by displaying video data on a screen during low light conditions.

Fig. 3. From left to right, there are monocular camera, stereo camera, and IR camera.

1.2.5 LiDAR

Lidar works by firing a laser beam at an object and then measuring the time taken for the light to bounce back to the sensor, to calculate the distance of an object. These systems can achieve high-resolution 3-D images and operate at longer ranges than camera systems. Some of the lidar scanners support surround-view sensors (that fire laser beams continuously in all directions), which can generate a

360° 3-D image of the surroundings with extremely accurate depth information. Lidar is becoming very popular in autonomous vehicles. Several prototype vehicles [7], [8] have demonstrated the advantages of using lidar in autonomous driving. Lidar is useful for systems implementing automatic braking, object detection, collision avoidance, and more. Depending on the type of sensor, lidars for cars can have a range of up to 60 m. Despite the aforementioned advantages, lidars are heavy, bulky in size, and expensive. Moreover, atmospheric conditions such as rain or fog can impact the coverage and accuracy of these systems. Emerging solid-state lidars [9] have opened the possibility of powerful lidars that are significantly smaller and relatively inexpensive.

1.2.6 RADAR

Radar systems emit microwaves and estimate the speed and distance of an object by measuring the change in the frequency of the reflected wave as per the Doppler effect. Due to the longer wavelength of microwaves, they can travel much farther than optical light (e.g., with lidar) and can detect objects at a longer distance. Unlike lidar, radar is not affected by foggy or rainy weather conditions and is relatively inexpensive. Depending on their operating distance range, radar systems can be classified as short range (0.2–30 m), medium range (30–80 m), or long range (80–200 m) [10] as shown in Figure 4. Cross-traffic alerts and blind-spot detection are some of the applications of short-/medium-range radars. These systems are often located
at the corners of a vehicle. Adaptive cruise control is a long-range radar application, with the system located behind the front grill or under the bumper. Researchers have been developing algorithms to improve the performance of radar and reliability all while attempting to reduce the cost and power of the system [11].

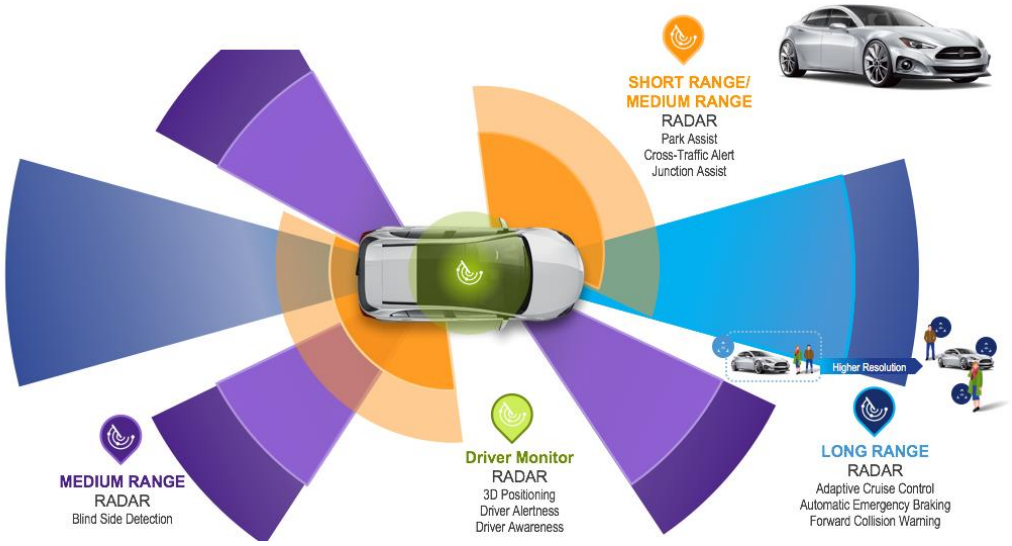

Fig. 4. Different ranges of RADARs used for different porpuses in autonomous driving.

1.2.7 Ultrasonic sensors

Ultrasonic sensors use sound waves to measure the distance to an object. These sensors are mainly used for detecting objects very close to the vehicle. Some example applications include automatic parking and parallel parking assist. These sensors are mainly located under the front and rear bumper of the vehicle.

The distance from the ground of a point of a vehicle body is computed as:

$$D = k.T_f.V_s \quad (1)$$

Where

T_f time of flight of an ultrasonic pulse, i.e., the time the pulse takes to cover the distance *D*;

k constant close to 0.5, which depends on the sensor geometry;

V_s velocity of sound in air.

The ultrasonic pulse is generated using a piezoelectric transducer and the echo reflected by the ground is received by another piezoelectric transducer. The two transducers are mounted close to each other to make up the measuring head. The uncertainty contribution due to the constant *k* can be made negligible by means of a sensor calibration after mounting the measuring head.

As the measured quantities T_f and V_s can be considered uncorrelated, the standard uncertainty $u(D)$ of the measured distance can be obtained from equation:

$$u(D) = \sqrt{\left(k.T_f\right)^2.u^2(V_s) + (k.V_s)^2.u^2\left(T_f\right)} \quad (2)$$

where $u(V_s)$ and $u\left(T_f\right)$ are the standard uncertainties of the velocity of sound and of the time of flight. The velocity of sound in air depends on the temperature and, to a lesser extent, on the air humidity *h:*

$$V_s = f(\theta, h) \quad (3)$$

therefore, (2) becomes

$$u(D) = \sqrt{(k.T_f)^2.\left[\left(\frac{\partial f}{\partial \theta}\right)^2.u^2(\theta) + \left(\frac{\partial f}{\partial h}\right)^2.u^2(h)\right] + (k.V_s)^2.u^2(T_f)} \quad (4)$$

If the humidity is considered a random variable uniformly distributed in the range of 10%RH to 90%RH, its effect on the velocity of sound is of about 0.15% at 20 C. This leads to a standard uncertainty contribution of about 0.3 mm for a distance range of 0.3 m, hence a humidity sensor is not necessary.

The velocity of sound in air depends on the temperature according to the approximated equation:

$$V_s \approx 20.055.\sqrt{T} \quad (5)$$

where is the absolute temperature, which is measured in kelvin.

Velocity-of-sound changes in the range of 330–360 m/s have to, therefore, be expected for temperature changes in the range of 0–40 C. Such an effect must be taken into account in the determination of the distance, hence a temperature sensor is re quired.

Another phenomenon that affects the uncertainty of the measured distance is the car speed, which has the same effect of the component of the wind that flows perpendicularly to the path of the ultrasonic pulse. Such effect consists in an increasing of the pulse path and, in turn, of the measured distance. As the maximum car-speed is of the order of 10% of the velocity of the sound, the distance error due to a car speed V_w can be approximately estimated as

$$\frac{\Delta D}{D} \approx \frac{1}{2}.\left(\frac{V_w}{V_s}\right)^2 \quad (6)$$

For a car speed of 33 m/s (about 120 km/h), the distance error at 0 °C ($V_s \approx 330\ m/s$) is of about 0.5%. One should note that this error could be easily corrected by the knowledge of the car speed.

Distance measurement in the range of 0.1 m to 0.3 m requires the measurement of time of flights in the range of 0.5–2 ms. The required distance standard uncertainty of 1 mm can be achieved by measuring the time of flight with a standard uncertainty of 2.5 μs, the temperature with a standard uncertainty of 1 °C, and avoiding the use of a humidity sensor.

Ultrasonic signals with frequencies in the range of 30 kHz to 5 MHz can be used to generate the pulse. Higher frequencies might be preferable since they imply lower wavelengths and thus a potentially better resolution, but the sound attenuation in the air dramatically increases as the frequency increases. In addition, higher frequencies require both costly transducers and fast electronic devices, therefore preventing a low-cost arrangement to be obtained. Lower frequencies have the advantage of low-scattering problems and can be obtained with low-cost transducers, but the wavelength in the air is several millimeters, thus requiring special care in order to obtain measurement uncertainties that are lower than the wavelength.

1.2.8 Others

A few other sensors are used to complement and improve the functionalities of those discussed earlier. For instance, photonic mixer device (PMD) cameras consist of an array of smart sensors that enable fast optical sensing and demodulation of incoherent light signals simultaneously [12]. PMDs can support parallel target pixel-wise distance measurement without scanning, thus resulting in faster imaging, high lateral resolution, and depth information. IMUs and GPSs are examples of systems that help improve the distance measurements with lidar and radar.

1.3 Vision-based ADASs

Vision-based ADASs rely on images from cameras and use computer vision principles to extract useful information.

1.3.1 Computer vision data flow for ADASs

Figure 5 shows the steps involved in a vision-based system, each of which is discussed.

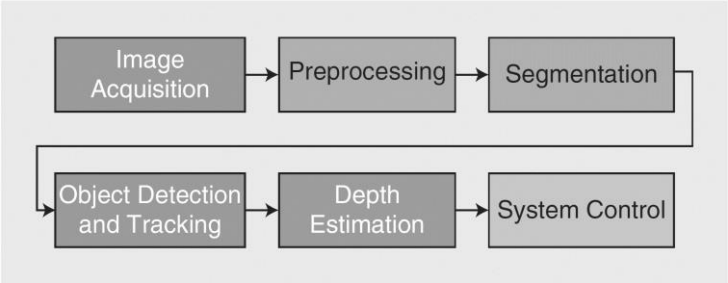

Fig. 5. The steps involved in a vision based system

1.3.2 Image acquisition

This refers to the process of capturing a frame from a video. The frame is often represented as a matrix of pixel data where each frame contains three channels of information, e.g., red, green, and blue (RGB) sets of pixels. Typical frame rates in ADASs range from five frames per second (fps) to 60 fps depending on the application. Applications that involve detection of vehicle proximity need a higher frame rate due to the rapid change in distance for cars on the road. In contrast, traffic sign detection does not demand a higher frame rate because only one frame of the sign needs to be captured for the sign to be detected.

1.3.3 Preprocessing

There are several common preprocessing steps needed to prepare an image for various computer vision algorithms, e.g., denoising, color enhancement, color space conversion, and image stabilization. A typical example of color space conversion is to convert the RGB color space to hue, saturation, and value to separate color from the intensity. Moreover, the hue channel is often used to separate out adverse effects (e.g., shadows, uneven lighting, and over- and underexposure) in the image to make tracking and detection easier.

1.3.4 Segmentation

This is the process of separating features from a frame. In analyzing an image, it is helpful to partition it into recognizable objects, e.g., identifying the road and sky in a frame as two different features. Various thresholding techniques are used to filter one class of pixels (e.g., the road) from another (e.g., the sky) as shown in Figure 6. One of the methods, e.g., exploits color information to detect a stop sign, where an algorithm may look for red in the image (typical for stop signs in the United States). Any pixels in that red range will be turned white, and anything that is not will be turned black. This results in a binary image that is often used as a mask for finding the area of interest on the original image.

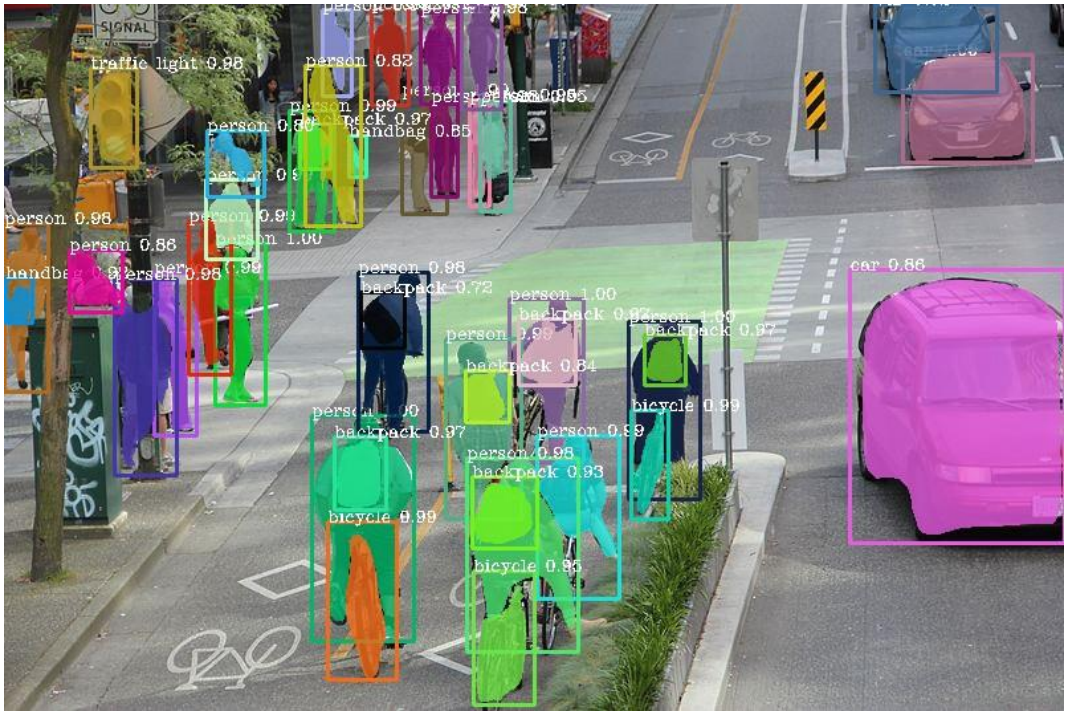

Fig. 6. The segmented objects that are categorized in different groups.

1.3.5 Object detection and tracking

This is the process of classifying an object in an image (e.g., determining if an object ahead is a vehicle, sign, or pedestrian) and predicting its movement. For instance as shown in figure 7, an object detection and tracking is done. It is often accomplished with various machine-learning (ML) algorithms. ML algorithms are provided large training data sets (thousands of images) to learn and differentiate between vehicles and common objects found around them. An example of an object detection method is called the cascade classifier, which was first presented in [13] for face detection, on low-performance hardware systems.

Another common technique to train and classify images is using a convolutional neural network (CNN), which typically consists of an input layer, multiple hidden layers, and an output layer. The hidden layers consist of convolution and pooling layers that are used for feature extraction and a fully connected layer for classification. Examples of CNN frameworks used for vision applications include Caffe, Darknet, and MATLAB. An application of a CNN for object tracking is discussed in [14]. Kalman-filter-based object tracking is proposed in [15], where the filter tracks the object's velocity.

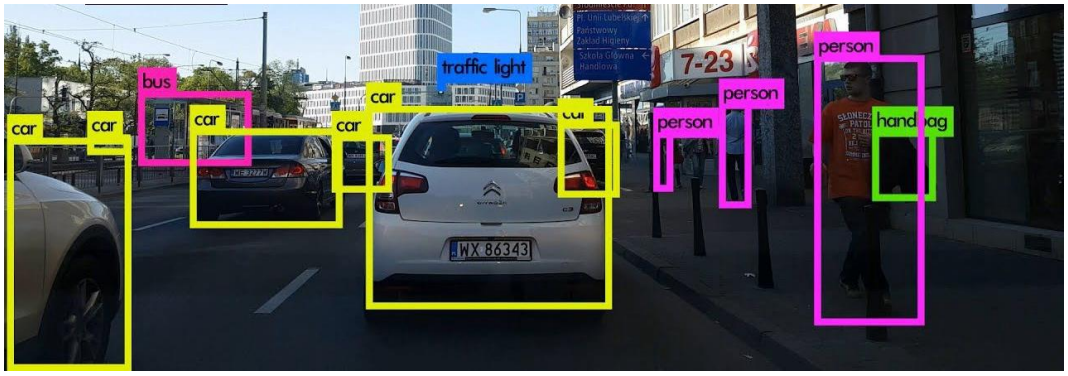

Fig. 7. An example of object detection and tracking.

1.3.6 Depth estimation

This step involves estimating the distance of an object in the image frame relative to the camera. There are two common methods for depth estimation: 1) the use of a stereo camera to create a stereo pair and develop them to make a depth map and 3-D point cloud that allow a real-world reconstruction of the scene [16]; and 2) the use of a monocular camera and several state-of-the art techniques that use a subset of optical flow, calibration, and least squares techniques [17]. An example of the depth estimation is shown in Figure 8.

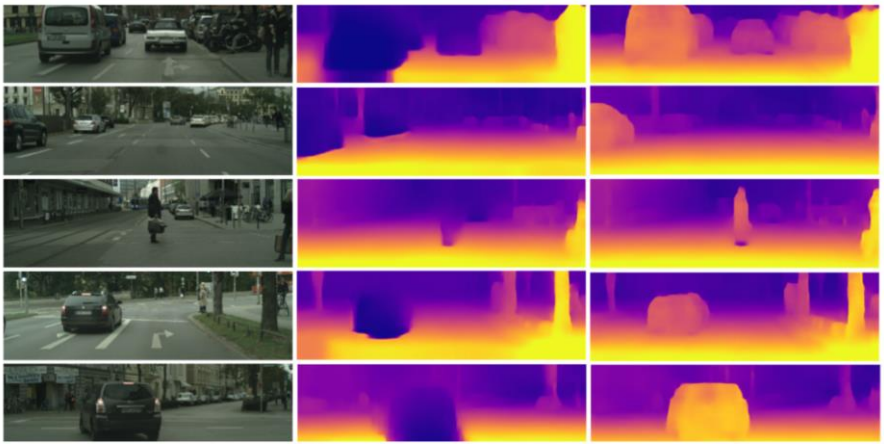

Fig. 8. Depth estimation helps understanding the distance of the different objects in images.

1.3.7 System control

This is the last step in the vision data flow, which involves interpretation of the outputs from previous layers. This step requires a weighting of each layer in the vision pipeline to come up with a confidence value that can be used to make decisions. A major challenge at this step is a false detection with high confidence that would take priority over other information obtained from the previous layers.

Thus, training with data that is correct and contains many orientations of the object to be classified is crucial to achieve high accuracy.

In the third chapter we introduce an Adaptive Model Predictive Control that recently is used in autonomous driving. Model Predictive Control uses the dynamic model of the system. For nonlinear systems we use AMPC. The name is Adaptive MPC since it adjusts the prediction model at run time.

1.4 Outdoor monitoring

In this section, we will discuss the classification of objects that are outside a vehicle, e.g., pedestrians, vehicles, and roads.

1.4.1 Pedestrian, Vehicle, sign, and lane detection

Detecting pedestrians is done using various classifiers, e.g., [18]. Often more than one classifier is used for detecting people because of the varying orientation and configuration in which pedestrians may appear. Deep-learning networks such as CNNs have been helpful to not only identify pedestrians but also classify their actions.

Vehicle detection is a major focus of object detection in ADASs. The fact that many vehicles share common features, such as having tires, brake lights, and license plates, allows the detection of these objects to indicate the presence of a car. These features are all used to distinguish the vehicle from other objects, such as signs, roads, and other miscellaneous objects. An example of vehicle detection is shown, using a CNN framework (Darknet) and a real-time detection system, You Only Look Once [19]. The orientation of vehicles can cause some issues with their identification. A vehicle being viewed from the front contains a different set of features than a vehicle from the side or back. Often vehicle classifiers consider various classes of vehicles, such as cars, trucks, and semis that are trained with many orientations.

Many ADASs are beginning to support traffic sign detection. The most common use case is determining the speed limit on the road by reading a speed sign (an ADAS would alert the driver if the vehicle speed is over the limit). For instance, color thresholds can be used to find the location of a sign and optical character recognition to determine what that sign displays. Other methods include using CNNs and hybrid techniques, such as [20].

Another ADAS feature used in a few production vehicles is the ability to keep the vehicle within the lane lines on the road. However, lane lines are one of the

hardest road features to detect because of their inconsistencies, such as being different colors, faded, and sometimes not even present. Current methods to detect lane lines often use a Canny transform to find the edges in the image. Once the edges are found, a Hough transform is used to compare the lines to a single slope to determine if they are indeed lane lines [21]. The use of CNNs is also becoming popular for lane line detection. When all the detection parts techniques gather together, we will have a detection of different objects of the world as shown in Figure 9.

Fig. 9. Object detection in different conditions of the roads.

1.4.5 Collision avoidance

ADASs are beginning to incorporate automatic braking and collision avoidance. This is done by combining many features discussed earlier, such as object tracking, vehicle detection, and distance estimation [14]. With this combination of data, a vehicle can predict a collision and stop it from happening by braking or even steering out of the way. A forward collision avoidance is shown in Figure 10.

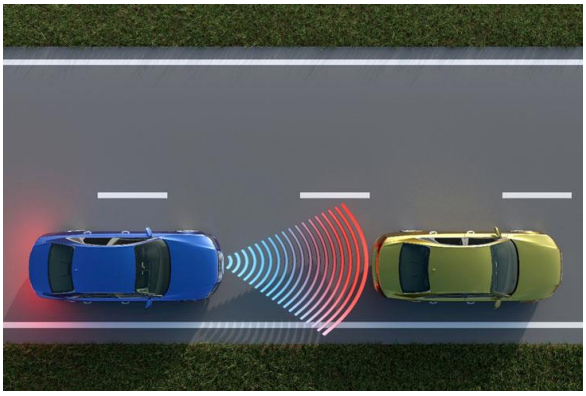

Fig. 10. Forward collision avoidance using Radar.

Sensor selection is the first and most crucial step towards designing a reliable and robust Forward Collision Avoidance. Active sensors perform well in different weather conditions and nighttime and their price is also in affordable range. The most common approaches to detect vehicles by active sensors include Radar-based and Laser or Lidar (Light Detection and Ranging) based. In the other hand, passive sensors with less common usage for Forward Collision Avoidance collect information by receiving the signals without emitting them and include acoustic and optical (camera) sensors.

1.5 Indoor monitoring

In a study conducted by the National Highway Traffic Security Administration [22], it was observed that driver fatigue, drowsiness, or distraction are the causes of 80% of vehicle accidents. As ADAS becomes prevalent in production vehicles, there has been an increase in focus on monitoring the driver using a camera pointed at him or her. If the driver accesses a phone or does not look at the road for a specific time duration, an alert or attempt to get off the road will be made [6]. Drowsiness-fatigue-detection systems have also included the ability to detect if the driver has fallen asleep and can attempt to alert the driver though a sequence of seatbelt vibrations and speaker alerts [23].

1.6 Next generation ADASs

Next-generation ADAS solutions are beginning to use sensor fusion and other advanced communication systems, such as vehicle-to-everything (V2X).

1.6.1 Sensor fusion

Sensor fusion refers to combining information from multiple homogenous or heterogeneous sensors to find a single best estimation of the state of the

environment. Fusion helps sensors complement each other's limitations and offers greater leverage to the system compared to a system with individual sensors. Sensor fusion offers high precision, reliability, robustness to uncertainty, extended spatial and temporal coverage, and improved resolution, which are crucial in safety-critical systems, such as vehicles. Although this comes at a higher computation cost, the computation power available in modern-day cars and the reducing cost of the sensors are facilitating the widespread integration of these systems.

The classification of different levels of sensor fusion along with the most commonly used techniques for fusing data are discussed in [24]. The growing interest in deep learning and other ML methods in recent years has driven researchers toward exploring more efficient and intelligent techniques that enhance ADASs with sensor fusion capabilities.

1.6.2 V2X communication

V2X communication represents a class of communication systems that provides the vehicle with an ability to exchange information with other systems in the environment. Examples include vehicle-to-vehicle (V2V) for collision avoidance, vehicle-to-infrastructure (V2I) for traffic signal timing, vehicle-to-network for real-time traffic updates, and vehicle-to-pedestrian for pedestrian signaling. State-of-the-art V2X communication is based on either dedicated short-range communications (DSRC) or cellular networks [25]. The IEEE 1609 family of standards for Wireless Access in Vehicular Environment (WAVE), which is developed based on the IEEE 802.11p standard, defines an architecture and a set of services and interfaces to enable DSRC-based secure V2V and V2I communication [26].

1.7 Autonomous vehicles

Next-generation ADASs using sensor fusion and V2X communication are paving the way for autonomous driving. The Society of Automotive Engineers (SAE) J3016 standard [27] defines six different levels of driving automation for on-road vehicles. A vehicle is categorized as level zero if there are no ADASs assisting the driver in handling steering and acceleration/deceleration and everything is handled manually by the driver. Level one vehicles consist of DASs assisting the driver in handling either steering or acceleration/deceleration under certain cases with human driver input. ADASs in level two vehicles handle both steering and acceleration/deceleration under certain environments with human driver input. In general, in lower-level vehicles (levels zero to two), the driver monitors the driving environment. In contrast, ADAS monitors the driving environment in higher-level (levels three to five) vehicles. Modern vehicles with the top-of-the-line ADASs, such as the 2016 Tesla model S, are level three, where multiple safety systems are handled by the system, but the driver intervenes when needed. Level four vehicles

handle multiple safety systems and operate in a wider range of environments. Level five automation is the end goal of autonomous driving, where all of the systems in the car are operated by the ADAS, under all driving conditions (such as snow-covered roads and unmarked dirt roads) and would not require any human intervention. This, however, still requires significant advances in multiple areas, such as sensor technology, computing systems, and automotive networks.

1.8 Challenges with ADASs

Despite significant advances in the field of ADASs, several important challenges remain to be overcome.

1.8.1 Changing environmental conditions

One of the major problems with today's ADASs is that the performance of the system is significantly impacted by changing environmental and weather conditions. For example, vision-based ADASs have issues with sensing during rainy and extreme lighting conditions (too dark and/or too bright) [28]. One of the possible solutions to this problem includes sensor fusion, by relying on other sensor data depending on the weather conditions, e.g., relying on the camera and radar during low light conditions while using the camera and lidar during other times for accurate distance estimation. The inclusion of V2I and developing cost-effective smart roads could help mitigate this issue.

1.8.2 Resource constrained system

Embedded systems used in ADASs have a requirement for low power consumption. This is because ADASs involve running several complex algorithms that result in high power consumption and thermal dissipation. Due to the limited availability of energy in vehicles, it is essential to minimize the power consumption of the embedded system used in ADASs.

Using more energy-efficient hardware than conventional general-purpose central processing units is important, which is why emerging ADAS hardware must rely on graphics processing units, digital signal processors, image signal processors, etc., that are customized to reduce power consumption for ADAS applications. Moreover, as the embedded systems for ADAS operate in real time, they have strict timing constraints, which establishes a latency minimization requirement. Hence, optimized hardware and software that results in minimal power consumption and greater performance (lower latency) predictability are highly desired in an ADAS.

1.8.3 Security

Modern vehicles are becoming increasingly connected with a lot of different systems, such as Wi-Fi, near-field communication, and V2X. This enables the vehicle to sense and receive a variety of information but also makes it more vulnerable to attacks. Many vehicle hacks have been demonstrated, e.g., researchers in [29] used onboard diagnostics (OBD-II) to hack a GM vehicle. In [30], the telematics system in a Jeep Cherokee was hacked to accelerate, brake, and kill the engine. This problem is aggravated in ADASs and autonomous driving. Preventing hackers from gaining access to connected vehicles is becoming increasingly important. This involves securing both in-vehicle networks and external communication.

1.8.4 Geospatial constraints

Many of the modern ADAS solutions being developed are tested within a geographic location or a group of locations where they are sold. This limits the ADAS to one or a certain group of geographical locations. This is because not all countries (or some states in a country) adhere to the same sign and road conventions uniformly, which makes ADAS algorithms that are often trained under one location hard to work efficiently in other locations. There is a need to improve algorithms, e.g., by exploiting V2X technology deployments to overcome variations in road sign conventions.

1.9 The developed ADASs

In this thesis, we tried to gather three important ADASs that we developed. The system that we developed started by concerning about the comfort driving and its most common comfort issue in autonomous driving, Motion Sickness. Considering the sources of the motion sickness, which will be discussed in the third and fourth chapters, we developed two motion sickness minimization methods that covers most of the sources. These systems, that will be discussed in the third and fourth chapters, are A Full-Featured, Enhanced Cost Function to Mitigate Motion Sickness in Semi- and Fully-autonomous Vehicles, Motion Sickness Minimization Alerting System Using The Next Curvature Topology. The research on Motion Sickness made us concentrate on the anticipating the next moves of the vehicle that is a common source of the motion sickness. To aim predicting the next movements, we started research on Motion Forecasting and soon we developed a methodology that outperform in the intersections. This system that will be discussed in the fifth chaper is Motion Prediction using Attention Heads and Traffic rules in intersections.

In the next chapters, after describing the high-performance embedded automotive platforms, we explain each ADAS we developed. There are five

potential sources of AV motion sickness; variation in horizontal and vertical acceleration, posture instability, loss of controllability and loss of anticipation of motion direction, Head downward inclination, and lack of synchronization between virtual motion and the vehicle motion profile [31]. First, A Full-Featured, Enhanced Cost Function to Mitigate Motion Sickness in Semi- and Fully-autonomous Vehicles [40] is explained as the state-of-the-art work that focuses on variation in horizontal and vertical acceleration and loss of controllability and loss and a control system is developed for it. Second, Motion Sickness Minimization Alerting System Using The Next Curvature Topology [41] is introduced that focuses on loss of controllability and loss of anticipation of motion direction and lack of synchronization between virtual motion and the vehicle motion profile. This system uses a Human Machine Interface (HMI) to alert the passengers. Finally, Motion Prediction using Attention Heads and Traffic rules in intersections is explained and this state-of-the-art work seeks to explain the improvements that can be done by adding the Traffic Rules into the Motion Prediction models.

Chapter 2

High-performance Embedded Automotive Platform

By developing the autonomous driving subsystem, the requirement of having novel embedded platforms with high speed and reliability increases. Each ADAS based on its functionality should be able to perform in real-time in the vehicle. As the complexity of the system increases, the need of the higher performance embedded platform rises. As discussed before, each ADAS needs fulfill some embedded platform capabilities and the systems that we designed also require a reliable setup for the implementation. In this chapter of the thesis, we investigate on the high-performance embedded automotive platforms and find the best choice for the systems we developed. We investigate how the different platforms perform and finally we discuss why we choose the embedded platform for our systems.

2.1 Introduction

There is a clear trend in the automotive domain towards a new paradigm of centralized Electrical/Electronic (E/E) architectures, where large portions of formerly separated functionalities running on dedicated electronic control units (ECUs) are integrated into centralized vehicle integration platforms (VIP) [32]. At the same time, novel computation, and data-intensive algorithms, such as, for instance, predictive maintenance or automated driving (AD) functionalities, are being deployed on these centralized high-performance platforms.

In order to satisfy the tremendous demand of “centralized” computing power, heterogeneous system on the chips (SoCs) are being increasingly deployed in automotive systems. These SoCs are microprocessor-based (μP-based), featuring a variety of integrated specialized accelerators, including graphics processing units (GPUs) and Field Programmable Gate Arrays (FPGAs). Examples of this class of SoCs include NXP’s S32V vision processor family, or the Tegra series offered by Nvidia.

Compared to traditionally used micro-controllers, these heterogeneous SoCs are highly parallel and feature complex memory systems, composed of multiple

levels of on-chip shared SRAM memories (caches or scratch pads) and off-chip DRAMs. Obviously, the increased complexity of the memory system that is shared between multiple execution engines on the SoC leads to a strong performance correlation between parallel executed applications [33]. While programmatically data is accessed transparently through virtual address spaces, it is physically stored at different (shared) memory locations, with different access latencies that are dynamically influenced by complex access and caching schemes as well as mechanisms for ensuring data coherency and consistency. For instance, in [33] it has been shown that the average (sequential) read access latency can vary by a factor of up to 8x on an Nvidia Tegra X1 platform.

In computer-science, a radical shift towards heterogeneous compute platforms is happening now, accelerated by the rise of Machine Learning and thus dedicated accelerators, and the plateauing of the Moore's law applied to CPU compute power. In the real-time computing landscape, this shift has given rise to high-performance real-time applications. On high performing, heterogeneous systems co-location of multiple mixed criticality workloads on the same SoC can dramatically improve the utilization of system resources, enabling resource sharing (e.g., IO devices, hardware accelerators, etc.) and improving the efficiency of data sharing across workloads.

However, co-location also comes at the cost of potential performance degradation, both average and worst-case, due to interference on shared resources, and increased uncertainty in terms of workload execution time. Both the academia and industry have been investigating the impact of shared resource contention on real-time and mixed critical software, on hardware requestors (e.g., CPU, GPU, other hardware accelerators) and on memory bandwidth availability, resources access latency, and jitter [34-36]. The advent of larger integrated platforms which will run real-time workloads alongside general-purpose operating system (GPOS) workloads now calls for those systems to being able to provision their resources in a quantifiable and predictable way. This becomes crucial to determine acceptable worst-case execution times (WCET) for real-time workloads and to ensure smooth and responsive operation of the GPOS workloads running alongside them.

To aid compartmentalize traffic streams on shared resources, silicon hardware designers and manufacturers have introduced, primarily in the infrastructure market, technology that allows memory transactions to be labelled and then subsequently confined to partitions of shared resources: Arm, MPAM [37], and Intel, CAT.

2.2 The performance

The traditional decentralized automotive E/E architectures are the result of multiple years of evolution of vehicle functionality. By using dedicated hardware for additional and possibly optional functionalities, decentralized architectures enabled and/or followed the distributed development paradigm between vehicle manufacturers and suppliers. They allow the structural partitioning of the vehicle system into functional domains. On the one hand, this is important for the planning, design and implementation of vehicle functionality in a parallel setup to minimize organizational interfaces. On the other hand, the corresponding functional partitioning (one function - one control unit) limits the functional interfaces and integration effects to the communication networks. In Figure 11, a regular E/E architecture with two embedded systems are shown.

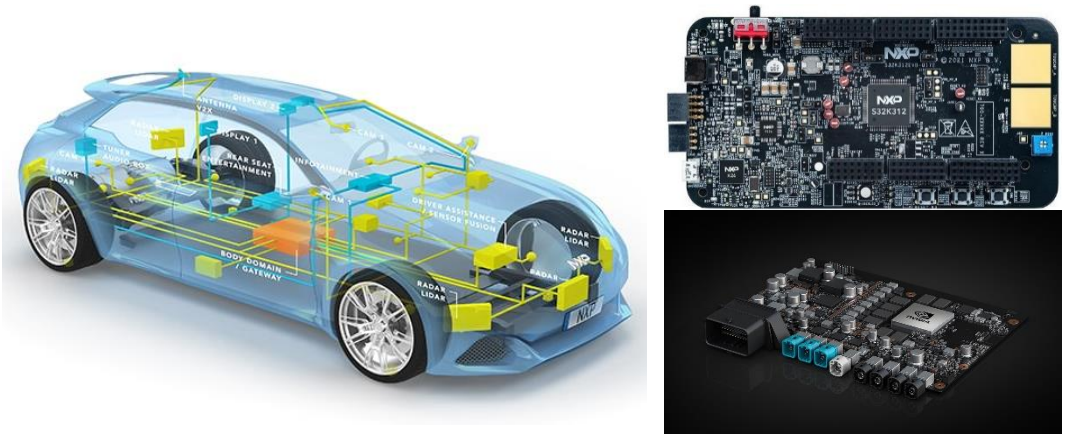

Fig. 11. From legt an example of E/E architecture of an intelligent vehicle and in the right two embedded platforms of NXP and Nvidia for developing the autonomous driving.

This architectural approach obviously results in a very close link between hardware and software since relocation of functionality is not an architectural driver. While decentralized architectures have carried the industry so far, new architectural drivers have appeared as automotive mega-trends: electrified, autonomous, connected and shared are the keywords that describe future expectations to a vehicle that must be backed by the E/E architecture. Centralized E/E architectures bring the opportunity of cost and weight savings by reducing the number of control units and promise to reduce complexity in comparison to a distributed E/E architecture. However, the complexity of managing distributed logic with dedicated resources is merely replaced by the complexity of managing centralized logic on a parallel hardware platform with shared resources [32].

In addition, these centralized control units need to host software categories which range from real-time safety-critical embedded software all the way up to ”app”-like software that come with the concept of being rather easily updated in field without negative side effects to other co-located functionality.

In this mixed-criticality setting, it is mandatory to have predictable performance and isolation of applications from each other, with respect to both space and time. This can be achieved by actively managing Quality of Service (QoS) and limiting the contention and interference on shared resources. Unfortunately, the currently available Commercial-Off-The-Shelf (COTS) platforms are rather optimized for high average performance and offer only coarse-grained support for configuring QoS for various shared resources, for instance, the interconnect or the DRAM. In order to achieve predictable performance, one has, thus, to resort to software-based methods. While spatial isolation is well supported, e.g. at the level of POSIX processes, several software measures have been introduced to limit the temporal interference on levels of scheduling, cache partitioning and memory bandwidth regulation. Scheduling is concerned with the distribution of CPU resources to applications. In comparison to the well-stablished priority-based scheduling approaches, reservation-based scheduling approaches show advantages in offering composable QoS guarantees to applications while allowing more flexibility than Time-division multiple access (TDMA)-based scheduling [38]. In general, partitioned scheduling, i.e., the pinning of application processes to cores, shows better predictability than global scheduling in multi-core settings as interference effects can be better localized. However, this approach has limitations as well, since in many SoCs the CPU cores are allocated in clusters of multiple cores (usually 2 or 4). These clusters provide shared infrastructure, e.g., the L2 cache. So, pinning a process on one core of a cluster will still not resolve the interference between cores of the same cluster on the L2 cache, unless that cache is partitioned. Extreme isolation mechanisms such as a ”stop the world” approach, where the execution of ASIL-D (Automotive Safety Integrity Level D) safety application on a single CPU core will stall all other cores in the system in order to generate a single-core equivalent scenario, are not adequate due to their performance penalty.

The previously mentioned issue of interference through caching can be addressed with cache coloring, exploiting the fact that (depending on the organization of the cache) certain address ranges will map to the same cache line. By choosing the mapping of virtual memory pages to physical pages with this in mind, performance-optimal memory allocation as well as cache partitioning can be achieved. However, this comes at the price of a factual smaller cache for each

partition and additionally fine-grained page-mapping that can cause side-effects in terms of page-table walks. Cache coloring can be supported by software on operating system or hypervisor level. Also, cache partitioning is directly supported by novel hardware (HW) mechanisms such as Arm DynamIQ as shown in Figure 12.

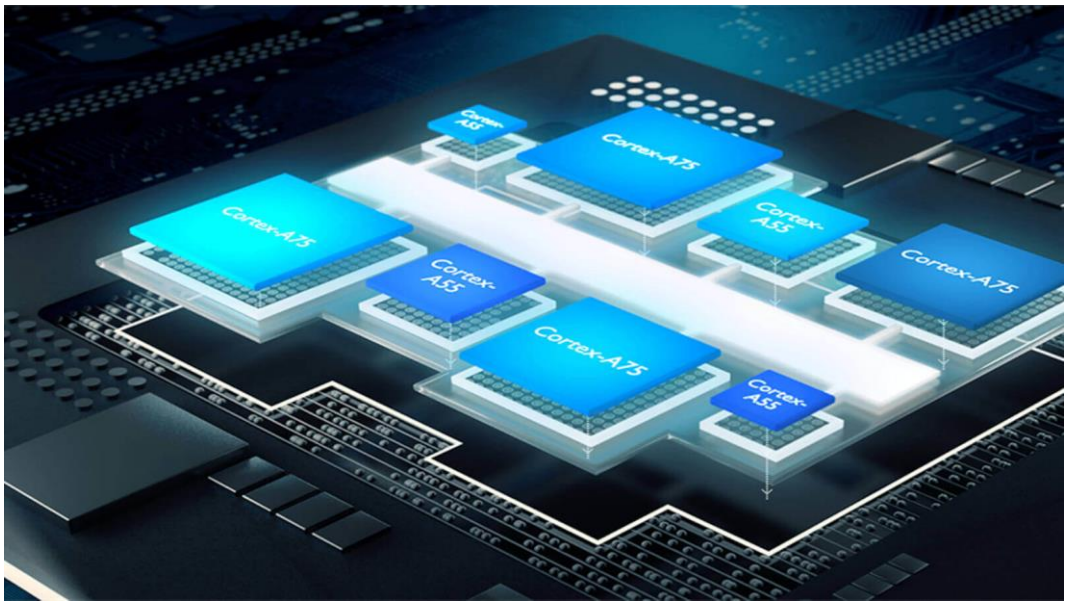

Fig. 12. Arm DynamIQ.

In order to address interference topics outside of a CPU cluster, e.g. the access to DRAM, performance counters integrated in the SoC can be used to actively limit the number of requests and reserve memory bandwidths at the level of cores, hypervisor partitions or single applications, using software based mechanisms such as Memguard [39]. This is an effective mechanism to limit interference. However, the more fine grained the objects to be isolated get, the higher the overhead becomes. This overhead could be reduced if the SoC exposed more information, e.g., the source of a particular request, or implemented less coarse resource partitioning mechanisms than in current SoCs (where QoS mechanisms are available at the cluster level, if at all) directly in HW.

All these concepts are sophisticated approaches with their individual drawbacks, such that their stand-alone configuration is already quite intricate for an industrial practitioner in real world application scenarios. However, there are additional interactions among these mechanisms. If you, e.g., use cache coloring to reserve cache for real-time critical applications in order to prevent cache thrashing

by non-real-time applications, you effectively reduce the cache size for all applications.

This could in turn lead to more DRAM traffic which will increase the DRAM interference also towards the real-time applications. Finding an optimal configuration for these interacting mechanisms is highly dependent on the characteristics of applications and the HW platform. Thus, automated profiling as well as sophisticated configuration tools are required. Considering updates in the field at operation time, it is absolutely crucial that there is as little human intervention required in this as possible.

In addition to these quantitative dependencies among these resources, the different resources (e.g., interconnect and memory) need also to be available at the same time in order to avoid interference due to resource contention.

Power consumption, performance (typically average or peak performance), and chip area are widely utilized design metrics considered when designing a computing system. Such metrics are typically obtained through measurement under a set of conditions representative of the intended system production deployment operations (platform target workloads). When designing real-time systems, additional performance metrics should be considered, such as quantifying how much the system allows confident computation of worst-case execution times (WCET) for each of the real-time workloads it is being designed to execute [32]. Typically, the degree of uncertainty on computing the WCET that characterizes current high-performance real-time compute platforms makes classical methods of computing the WCET unfeasible (such as analytical) [39]. Therefore, there would be the adoption of the following empirical performance metrics: i) Worst-case measured performance and ii) Time-predictability, defined as the quotient between the best-case measured performance and the worst-case measured performance.

2.3 Sources of uncertainty

The reason for high uncertainty in determining the WCET is typically down to specific sources of uncertainty. The sources of uncertainty in the following affect the ability to predict or even precisely measure the timing characteristics of real-time systems:

- Workload input data or events: they cause uncertainty when influencing the software control flow or the amount of computation performed by it. In this case it is said that the workload is data-variant. For example, conditional

branches based on values provided by or calculated from input data can lead to different paths of execution. Also, the depth of loops or recursions may depend on the content or size of the input data.

- Hardware state: state of the hardware resources at beginning of execution. Examples are initial cache contents or memory controller row buffer contents.
- Interference: deviation in performance caused by workloads that contend for the same shared resources, alter the initial hardware state for other workloads or both.

2.4 Shared resources and interference channels

As interference arises from contention between workloads, on accessing or using shared resources, co-location of workloads on high-performance system is prone to be affected by such contention, which calls for its accurate quantification. Each hardware shared resource can exhibit one or more interference channel, each one corresponding to a place in the resource where a specific type of contention can happen. The following are examples of potential resource interference channels:

• Internal hardware buffers between pipeline stages: a congested buffer may result in a general resource stall, delaying the service provided by the resource.

• Arbitration policies: they govern which workload has access to the resource at any given time. Biased policies (e.g., strict priority ones) or generally non-work-conserving ones can cause starvation of workload request flows.

2.5 NVIDIA GPUs

The graphics processing unit (GPU), first invented by NVIDIA in 1999, is the most pervasive parallel processor to date. Fueled by the insatiable desire for life-like real-time graphics, the GPU has evolved into a processor with unprecedented floating-point performance and programmability; today's GPUs greatly outpace CPUs in arithmetic throughput and memory bandwidth, making them the ideal processor to accelerate a variety of data parallel applications.

Efforts to exploit the GPU for non-graphical applications have been underway since 2003. By using high-level shading languages such as DirectX, OpenGL and Cg, various data parallel algorithms have been ported to the GPU. Problems such as protein folding, stock options pricing, Structured Query Language (SQL) queries, and MRI reconstruction achieved remarkable performance speedups on the

GPU. These early efforts that used graphics application programming interfaces (APIs) for general purpose computing were known as general-purpose computing on graphics processing units (GPGPU) programs.

While the GPGPU model demonstrated great speedups, it faced several drawbacks. First, it required the programmer to possess intimate knowledge of graphics APIs and GPU architecture. Second, problems had to be expressed in terms of vertex coordinates, textures and shader programs, greatly increasing program complexity. Third, basic programming features such as random reads and writes to memory were not supported, greatly restricting the programming model. Lastly, the lack of double precision support (until recently) meant some scientific applications could not be run on the GPU.

To address these problems, NVIDIA introduced two key technologies—the G80 unified graphics and compute architecture (first introduced in GeForce 8800®, Quadro FX 5600®, and Tesla C870® GPUs), and Compute Unified Device Architecture (CUDA), a software and hardware architecture that enabled the GPU to be programmed with a variety of high level programming languages. Together, these two technologies represented a new way of using the GPU. Instead of programming dedicated graphics units with graphics APIs, the programmer could now write C programs with CUDA extensions and target a general purpose, massively parallel processor. This new way of GPU programming is called "GPU Computing"—it signified broader application support, wider programming language support, and a clear separation from the early "GPGPU" model of programming.

NVIDIA GPUs increase in complexity at each newer generation. Gaining a deep understanding of GPU memory hierarchy as they evolve is necessary to write efficient code. It is especially important to know the size of each cache memory level, whether that memory is co-located with another cache that might evict its contents, and whether each cache memory is private to a streaming multiprocessor or shared among all. Figure 13 shows one example of the NVIDIA GPUs.

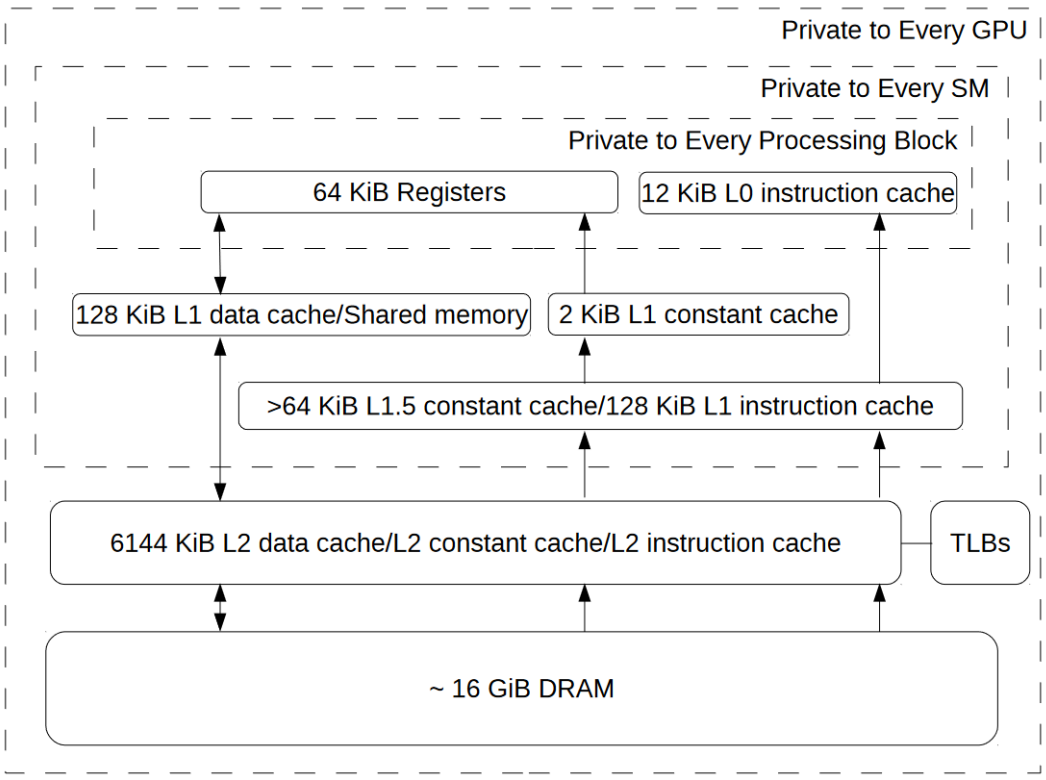

Fig. 13. The volta's architecture

As published by NVIDIA, the V100 GPU employs HBM2 memory, which offers a bandwidth of 900 GB/s (at 877 MHz), in conjunction with a L2 cache of 6,144 kibibyte. Data loaded from global memory is implicitly cached in L1 and L2.

NVIDIA introduced several architecture as Hopper, Ampere, Turing, Volta, Pascal, Kepler, Maxwell, and Fermi.

2.5 The conclusion

As per the systems that we developed, we needed an embedded platform based on our needs to be powerful and high performance. The target embedded platform, NVIDIA Jetson AGX Xavier, shown in Figure 14, is representative of the next-generation AV Domain Controller. This platform with a GPGPU of 512-core Volta with Tensor Core and a CPU of ARM 8-core v8.2 64-bit is an appropriate choice for the AD systems.

Fig. 14. Nvidia AGX-Xavier

The NVIDIA® Jetson AGX Xavier™ module delivers up to 32 TOPS of accelerated computing capability in a compact form factor consuming under 30 Watts. This gives you more than 20X the performance and 10X the energy efficiency of its predecessor, the NVIDIA Jetson™ TX2.

This advanced system-on-module is powered by the NVIDIA Xavier SoC and designed specifically for autonomous machines. Heterogeneous accelerated computing architecture delivers advanced edge capabilities. Plus, it comes with integrated memory, storage, power management, and an innovative thermal design to enable faster time to market. Run modern AI workloads and solve problems in areas like manufacturing, logistics, retail, service, agriculture, smart cities, and healthcare.

Jetson AGX Xavier is supported by NVIDIA JetPack, which includes a board support package (BSP), Linux OS, NVIDIA CUDA®, cuDNN, and TensorRT™ software libraries for deep learning, computer vision, GPU computing, multimedia processing, and much more. It’s also supported by the NVIDIA DeepStream SDK, which delivers a complete toolkit for real-time situational awareness through intelligent video analytics (IVA). This helps you boost performance and accelerate software development, while reducing development cost and effort.

Chapter 3

A Full-Featured, Enhanced Cost Function to Mitigate Motion Sickness in Semi- and Fully-autonomous Vehicles [40]

Current full- and semi- Autonomous car prototypes increasingly feature complex algorithms for lateral and longitudinal control of the vehicle. Unfortunately, in some cases, they might cause fussy and unwanted effects on the human body, such as motion sickness, ultimately harnessing passengers' comfort, and driving experience. Motion sickness is due to conflict between visual and vestibular inputs, and in the worst case might causes loss of control over one's movements, and reduced ability to anticipate the direction of movement. In this chapter, we focus on the five main physical characteristics that affect motion sickness, including them in the function cost, to provide quality passengers' experience to vehicle passengers. We implemented our approach in a state-of-the-art Model Predictive Controller, to be used in a real Autonomous Vehicle. Preliminary tests using the Unreal Engine simulator have already shown that our approach is viable and effective, and we implemented and evaluated using Motion Sickness Dose Value and Illness Rating and then tested it in an embedded platform. We implemented it on our embedded platform, NVIDIA Jetson AGX Xavier that is representative of the next-generation AV Domain Controller.

3.1 Introduction

In semi- and full AVs, vehicle control shall consider passengers' stress, and not decrease their level of comfort [42]. It was proven that a tight relationship exists between comfort and trust, as well as the acceptance of automated vehicles [43].

The mostly known comfort issues for the passengers is probably Motion Sickness. Its common symptoms are: headache, pallor, sweating, nausea, vomiting, and disorientation, and they can be measured by Physiological signals, Vestibule

Ocular Reflex (VOR) parameters, and Posture stability. There are several ways to mitigate this, such as instance visual cues, Posture and vehicle controllability, and Immersive Experience [31].

Motion is primarily sensed by the organs of balance located in the inner ear and our eyes, which are mainly or uniquely sensitive to accelerations. The vestibular section of the inner ear is partly comprised of three semi-circular canals that detect head angular acceleration. The main issue stems from the fact that our bodies are not used to low-frequency oscillating motion, and our "biological IMUs" are highly sensitive to this. In carsickness, the lateral accelerations (sway) in the low-frequency bands (0.1-0.5 Hz) are most relevant and their effects increase in higher accelerations. In general, researchers proved [44] that it might happen when the frequency is below 1 Hz.

The potential sources of AV motion sickness can be divided into five groups, namely, are variation in horizontal and vertical acceleration, posture instability, loss of controllability and loss of anticipation of motion direction, Head downward inclination, and lack of synchronization between virtual motion and the vehicle motion profile [31]. Although motion sickness is most frequently caused by a conflict between visual and vestibular inputs, loss of control over one's movements and reduced ability to anticipate the direction of movement are also important in the etiology of motion sickness [45]. All three factors, to varying degrees, are more frequently experienced by vehicle passengers than by drivers, who rarely experience motion sickness [45]. Possible countermeasures can be categorized into two groups: prevention solutions and mitigation solutions. Roughly speaking, the degree of motion sickness may be predicted by an acceleration frequency weighting that is independent of frequency from 0.0315 to 0.25 Hz and reduces at 12 dB per octave (i.e., proportional to displacement) in the range 0.25 to 0.8 Hz [31].

We contribute to research with the original design of a control software component for AVs that minimizes the most important costs. Model Predictive Control (MPC) is a reference framework for vehicle control because it includes both kinematic and dynamic models on the vehicle in its formulation. For this reason, MPC-based Advanced Driver Assistance Systems and Autonomous Vehicles are important research directions for mitigating road accidents. The real-time trajectories based on the pre-defined model might not be optimal. Therefore, an adaptive MPC design with an on-line vehicle parameter estimator is needed to account for those unpredictable changes. In this chapter, we used an Adaptive

Model Predictive Control (AMPC) that can estimate and update the model in real-time along with five constraints to build our cost function to minimize.

We have chosen the constraints that are relevant to Motion Sickness and comfort, either direct (acceleration frequency, longitudinal acceleration, and lateral acceleration) or indirect (speed limitation and distance from the next vehicle). Acceleration frequency is one of the constraints that directly affects the Motion Sickness and the range of Frequency in which Motion Sickness occurs in acceleration frequency between $0.0315<F<0.8$ Hz [46] so we need to avoid this range. Speed limitation is not directly related to Motion Sickness level. However, In the other hand, as the speed goes up, Acceleration Frequency for the speed regulation will arise. Therefore, we consider a speed limitation based on our Acceleration Frequency. The European New Car Assessment Program (Euro NCAP) performed standardizing tests on different autonomous vehicles with a constant speed of 20 – 60 km/h [47]. However, we challenge the work with the speed between 0-80 km/h. We also consider a threshold of acceleration because it affects both Motion Sickness and Comfort driving [47]. It is also one of the factors that increase Motion Sickness Dose Value (MSDV). Therefore, having the limitation with an appropriate planner can lower the MSDV and raise comfort. We also consider the distance from the next vehicle to brake with a minimum acceleration, as we discussed before. In particular, with higher distance from the next vehicle, we require a lower braking acceleration. Finally, since the lateral acceleration is the other important source in MSDV [46], we need a lane keeper to reduce our lateral accelerations to a minimum quantity.

The system is tested on MATLAB/Simulink [48] and then implemented on an NVIDIA Xavier AGX. We evaluate our work based on ISO 2631-1 [47] which a measure of the probability of nausea that is called motion sickness dose value (MSDV) and a simple linear approximation between MSDV and mean passenger named illness rating (IR) are considered as the evaluation methods.

In the following sections, we first review the state-of-the-art in motion sickness and MPC controller. Then we describe the details of our controller. Finally, we show our implementation, and discuss experimental results with respect to the reference metrics of motion sickness.

3.2 Motion sickness in AV literature

In the recent years some efforts have been done to mitigate and minimize the motion sickness. These works can be categorized in two different groups. The first group tries to minimize the MS by having a new motion planner with a library of costs. In this regard, In [40], five different main physical characteristics that can be effective on motion sickness, and defining them in a function cost, to improve quality passengers' experience and minimize the Motion Sickness to vehicle passengers is considered. In [42], the costs of consisting of progress, comfort, and safety are utilized for the evaluation of the strategies generated by the three modules of distance keeper, lane selector, and merge planner. In [45], on investigation with two strategies for decreasing the visual-vestibular conflict while watching videos is conducted. The first approach locates visual stimuli on or around the video screen to mimic the perceived motion and forces of the moving vehicle. The second method tries to control the position of displayed images synchronized with passenger's head motions produced by vehicle acceleration/deceleration and vehicle motions, then provides a video that appears to be stabilized in relation to the movement of the vehicle. In [52], they generate the optimal Path Planning using Clothoid Curves to increase the comfort of the passengers. They use the second clothoid length, the straight line to the goal at the end, made up of the first clothoid length, and the squared distance along the curve as their costs to control. To minimize the MSDV in autonomous vehicles, [53] presents an application of motion planning [53]. On the other hand, in the second group, the researchers try to have a anticipation alert to the passengers, so their brain will be ready to start the maneuvers. In this regard, in [54], they investigate the effects of peripheral information about upcoming maneuvers through a vibrotactile display in increasing the fully-automated driving car passengers' awareness of situations and mitigating their motion sickness level. This study concludes that in order to mitigate motion sickness inside a fully-automated driving car, more specific information need to be included in the peripheral information. In [55], they have progressed a prototype of a human– machine interface (HMI) that presents anticipatory ambient light cues for the AV's next turn to the passenger. The HMI prototype was proven to be effective regarding highly susceptible users. In [56] average illness ratings were significantly lower for the condition that contained informative auditory cues, as compared to the condition without informative cues. One second in advance of each displacement a sound clip was played over headphones communicating either "forward" or "backward" in the native language of the participant. In addition, recently, [57] resulted that if there is an additional effect of augmented visual stimulion MS, the effect is at best small. Therefore, having an augmented visual stimulion is not in our plan.

Although using different methods can lower the MS level, most of them are not tested in a real autonomous vehicle. Furthermore, the sound cues should be in a way that the passengers do not disturb. Indeed, the visual cues should be in a way that

shows the regular view of the vehicle not an augmented one [57]. In this regard, we focus on anticipatory audio and video cues using pleasant sounds and a Human Machine Interface to display and inform the passengers about the upcoming trajectories that may lead to make the passengers sick. To be able to anticipate the next moves, we require an evaluation system of the next 500 meters of the road using the map. The road is investigated based on the amount of the turns and the maximum speed allowed that lead to lateral accelerations that is high enough based on Motion Sickness Dose Value to make the passengers sick. The system alerts the passengers through a Human Machine Interface to focus on the road for prevention of the Motion Sickness.

Considering recent AMPC implementations in autonomous driving, concentrating on their cost functions, there are several efforts. In [58], an adaptive model predictive control with three constraints, Lane Change-Related Constraint, Location in Opposite Lane Constraint, and Maneuver Completion, is applied for tracking the references being generated for the Autonomous Vehicles on Two-Lane Highways. In [59], they constructed an adaptive model predictive control trajectory tracking system with the four constraints that define as follows: (1) The radius of all planned paths should be greater than the minimum turning radius of the wheel loader; (2) The planning path and its curvature should be continuous to provide steering stability of the loader; (3) When the loader is at the loading and unloading points, the articulation angle should be as close to zero as possible to avoid rollover of the vehicle; (4) The maximum planned velocity of the loader should not exceed 3m/s and rapid acceleration and deceleration should be avoided. In [60], an adaptive model predictive control (AMPC) scheme is developed to improve the yaw stability for four-wheel-independently actuated electric vehicles by minimizing the total longitudinal forces of all wheels. In [61], the side slip angle of the centre of mass and the side slip angle of the tire as hard constraints and the lateral acceleration as a soft constraint are considered to propose an Adaptive Model Predictive Control for Uncertain model (UMAMPC) algorithm to predict control variables for the next sampling time and alleviate the target angle discontinuity. In [62], they develop a fault tolerant path tracking control algorithm through combining the adaptive model predictive control algorithm for lateral path tracking control and Kalman filtering approach with two states chi-square detector and residual chi-square detector for detection and identification of sensor fault in autonomous vehicles by using the incremental constraint of tire and the incremental constraint of lateral acceleration.

In all of the above works, that are proposed for controlling the autonomous vehicles by AMPC, below than five constraints are used. In this chapter of the

thesis, we use five constraints in an AMPC that minimize the MSDV with consideration of comfort.

3.3 Control system

To design the controller, we defined a Vehicle Model and used the tire forces to specify our state space. Then, we entered our state space in AMPC and defined our constraints in it.

3.3.1 Vehicle Model

For an MPC control design, we require to define our Vehicle Model. It was found that the vehicle side slip angle is less than 1◦ in the highway autonomous or manoeuvre driving under clothoid constraints [63]. Thus, it is considered that the tire slip angle is also negligible under highway driving conditions, including cases employing an advanced driver assistant system (ADAS). It makes it possible to use a standard dynamic "bicycle model" [64] to describe the Vehicle Dynamics. Such as a recent work [65] that uses the higher speed until 35 m/s (126 km/h) with a bicycle dynamic model, we use a bicycle dynamic model for our tests between the speed of 0 km/h to 80 km/h and we use them in our first scenario. In the bicycle model, the two left and right wheels are represented by one single wheel. The model is derived assuming both front and rear wheels can be steered by δf and δr angles and the distances of front and rear wheels are a and b. The model neglects roll and pitch motions. The Motion of the vehicle is represented by X, Y and ψ. Figure 15 depicts a diagram of the vehicle model, which has the following longitudinal, lateral, and turning or yaw equations:

$$m\ddot{x} = mr\dot{y} + 2F_{x,f} + 2F_{x,r} \quad (1)$$

$$m\ddot{y} = -m\dot{x}\dot{\psi} + 2F_{y,f} + 2F_{y,r} \quad (2)$$

$$I_{zz}\ddot{\psi} = 2aF_{y,f} - 2bF_{y,r} \quad (3)$$

The vehicle's equations of motion in an absolute inertial frame are

$$\dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi \quad (4)$$

$$\dot{X} = \dot{x}\cos\psi - \dot{y}\sin\psi \quad (5)$$

The following equations hold for rear and front axes by using the corresponding subscript for all the variables (it is correct either for δ_f or δ_r). Longitudinal and lateral tire forces lead to the following forces acting on the center of gravity:

$$F_y = F_l \sin\delta + F_c \cos\delta, \quad (6)$$

$$F_x = F_l \cos\delta - F_c \sin\delta. \quad (7)$$

Tire forces for each tire are (accordingly α can be α_r or α_f)

$$F_l = f_l(\alpha, s, \mu, F_z), \quad (8)$$

$$F_c = fc(\alpha, s, \mu, F_z), \quad (9)$$

where α is the slip angle of the tire and s is the slip ratio. The tire model is considered as indicated in [66] velocities, respectively, are expressed as

$$v_{l,f} = v_{y,f} \sin\delta_f + v_{x,f} \cos\delta_f, \quad (10a)$$

$$v_{c,f} = v_{y,f} \cos\delta_f - v_{x,f} \sin\delta_f, \quad (10b)$$

$$v_{l,r} = v_{y,r} \sin\delta_r + v_{x,r} \cos\delta_r, \quad (11a)$$

$$v_{c,r} = v_{y,r} \cos\delta_r - v_{x,r} \sin\delta_r, \quad (11b)$$

And

$$v_{yf} = \dot{y} + a\dot{\psi} \qquad v_{yr} = \dot{y} - b\dot{\psi}, \quad (12)$$

$$v_{xf} = \dot{x} \qquad v_{xr} = \dot{x}. \quad (13)$$

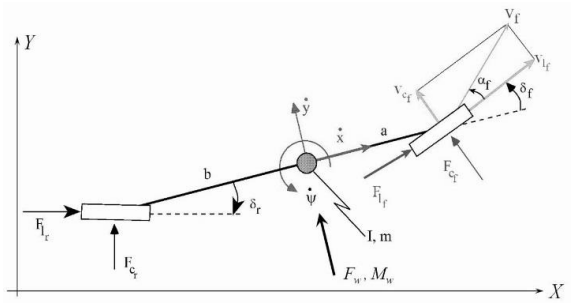

Fig. 15. Bicycle Model of the Vehicle

If we consider δr =0, then:

$$\ddot{x} = r\dot{y} + \frac{F_{l,f}cos(\delta_f) - F_{c,f}sin(\delta_f) + F_{l,r}}{m} \quad (14)$$

$$\ddot{y} = -r\dot{y} + \frac{F_{l,f}sin(\delta_f) + F_{c,f}cos(\delta_f) + F_{l,r}}{m} \quad (15)$$

$$\dot{r} = \frac{a(F_{l,f}sin(\delta_f) - F_{c,f}cos(\delta_f)) - bF_{c,r}}{Izz} \quad (16)$$

Using the equations (1)-(16), the nonlinear vehicle dynamics will have the states of $[\dot{X} \quad \dot{Y} \quad r \quad \dot{v}x \quad \dot{v}y \quad \dot{r}]$.

3.3.2 Adaptive Model Predictive Control System

MPC [67] is a method for process control that actively uses the dynamic model of the system. If the nonlinearity is high, however, MPC performance could deteriorate. In this case, one can use an AMPC that constantly predicts the new operating conditions [68].

An adaptive MPC algorithm is designed by using the recursively-identified state-space models with dynamic adjustments of MPC constraints and objective function weights [69]. Adaptive MPC controllers adjust their prediction model at run time to compensate for nonlinear or time-varying plant characteristics. Furthermore, Adaptive control for constrained systems has mainly focused on improving performance with the adapted models, while the constraints are satisfied robustly for all possible model realizations and the worst disturbance bounds [70]. In this chapter of the thesis, we used an Adaptive MPC to update our state-space online and get the linear part of our nonlinear system. This approach is implemented with the most important costs that we wanted to control.

In AMPC, the controller uses the time-varying Kalman filter (TVKF) instead of the static one to provide consistent estimation with the updated plant dynamics. The TVKF approach can be expressed as follows [102]:

$$L_K = \left(A_k P_{k|k-1} C_{m,k}^T + N\right)\left(C_{m,k} P_{k|k-1} C_{m,k}^T + R\right)^{-1}$$

$$M_K = P_{k|k-1} C_{m,k}^T \left(C_{m,k} P_{k|k-1} C_{m,k}^T + R\right)^{-1} \quad (17)$$

$$P_{k|k+1} = A_k P_{k|k-1} A_k^T - \left(A_k P_{k|k-1} C_{m,k}^T + N\right) L_k^T + Q$$

In equation (17), Q, R, and N matrices are constant covariance matrices, and A_k and $C_{m,k}$ are matrices depicting the state-space description of the system. The $P_{k|k-1}$ is the state estimate error covariance matrix at k constructed from the information from time $k-1$. TVKF is constructed to update regularly the L and M matrices with the updated plant dynamics.

3.3.2.1 Constraints

The Model Predictive Control can directly include constraints in the computation of the control moves which leads to linear program (LP) or quadratic program (QP)

to be solved at each sampling instance, with the constraints written directly as constraints in the LP/QP.

The MPC algorithm solves a quadratic optimization problem at each time interval. The solution of the problem determines the so-called manipulated variables (MV), which are essentially the input variables adjusted dynamically to keep the controlled variables (CV) at their set-points. The AMPC approach follows the same cost optimization algorithm as MPC with the cost function

$$J_y(z_k) = \sum_{j=1}^{n_y} \sum_{i=1}^{p} \left\{ \frac{w_{i,j}^{j}}{s_j^y} \left(r_j(k+i|k) - y_j(k+i|k) \right) \right\}^2 \qquad (18)$$

where k represents the current control interval, p is the prediction horizon (interval number), n_y is the number of plant output variables, z_k is the quadratic problem (QP) selection which is depicted as the formula $z_k^T = [u(k|k)^T \; u(k+1|k)^T \ldots u(k+p-1|k)^T \; k]$, $y_j(k+i|k)$ is the jth CV at the ith prediction horizon step, $r_j(k+i|k)$ is the ith references variable at the ith prediction horizon step, s_j^y is the scale factor for the jth plant output variable, and $w_{i,j}^{j}$ is the tuning weight coefficient reflecting the relative importance of the plant output variable. Among these variables n_y, s_j^y, p, and $w_{i,j}^{j}$, are determined during the controller design and stay constant.

Acceleration Frequency.

The frequency range of the tested Motion Sickness is 0.0315<F<0.8 Hz and this is very important to mention that the maximum Motion sickness occurs at 0.2 Hz [46]. Therefore, we tried to fix the frequency at 0.2 Hz (or T= 5s). In the other word, we try to prevent inserting acceleration in the period of 5 seconds.

Speed limit.

As discussed, the test speed is in the range of 20 – 60 km/h [47]. Since we need to consider having acceleration and braking in our work, we raised this limitation to 0 - 80 km/h and in our tests, we consider these values.

Acceleration limit.

Acceleration limitation is an important source for comfort and the different level of comfort is measured based on it [47]. Based on ISO 2631 [47] for determination of acceleration, the best range of the acceleration is <0.315 m/s^2 that is named not uncomfortable. In this standard, the best range of acceleration is <1 m/s^2 that is fairly uncomfortable, and it is the border of the uncomfortable range of measurements. So we maintain this range.

Distance to the front vehicle

With a higher distance from the next vehicle, we decrease the braking acceleration. It means that we will have more time to plan smooth braking, with the consideration of our acceleration limit, and it lowers the MSDV. There is a Two-Second Distance rule from the next vehicle [71]. The mean deceleration is 2.5 m/s^2 [72] and our deceleration should not exceed 1 m/s^2. Therefore, we raised the distance to Five-Seconds Distance to fulfil these requirements.

Lane keeper

As discussed, we have high importance in lateral acceleration to minimize the MSDV. Therefore, our system maintains the boundaries and controls the Y as the centre of the road lines. It is obtained by having a reference Y of the road and try to follow it. In the results, we show that our controller follows it properly.

3.3.3 Motion Sickness Evaluation

The total MSDV resulted from lateral and longitudinal motion is given as [47]:

$$\text{MSDV}=\sqrt[2]{\int_0^T (a_{x,w}\,(t))^2} + \sqrt[2]{\int_0^T (a_{y,w}\,(t))^2} \qquad (19)$$

Where $a_{x,w}(t)$ and $a_{y,w}(t)$ are the frequency weight acceleration in the longitudinal and lateral direction.

$$a_{x,w}(t) = a_x(t) \times W_f \qquad (20)$$

$$a_{y,w}(t) = a_y(t) \times W_f \qquad (21)$$

where $a_x(t)$ and $a_y(t)$ are the longitudinal and lateral acceleration. W_f is the weighting factor defined in British Standard 6841 [47] for evaluating low frequency motion with respect to motion sickness. From the standards [47], [73], a simple linear approximation between MSDV and mean passenger illness rating is given as:

$$\text{IR} = \text{K} \times \text{MSDV} \qquad (22)$$

where IR is predicted illness rating and K is an empirically derived constant. The illness rating value is divided into four levels; 0 indicates feeling fine, 1 indicates slightly unwell, 2 indicates quite ill, and 3 indicates absolutely dreadful [47], [73].

3.4 Implementation

The system was tested in MATLAB/Simulink [48] and then implemented by an NVIDIA Xavier AGX. This platform is representative of next-generation AV Domain Controller where AD software components, such as our controller, will execute.

To verify the validity of the proposed AMPC controller. CarSim [49] is used to provide a vehicle dynamics model and MATLAB/Simulink is mainly for providing control function.

Two different scenarios, straight and turn, were tested. The scenarios were designed in drivingScenarioDesigner and tested by using Unreal Engine [50] for the visualization of the output.

3.4.1 Scenarios

Since the MSDV is mainly a result of the lateral and longitude accelerations, we require to define the scenarios based on the existence of longitudinal acceleration, braking, and lateral acceleration. Therefore, we define a straight scenario that has the longitudinal acceleration and braking, and a turn scenario that has longitudinal and lateral accelerations.

3.4.1.1 Straight road

In the straight scenario, we made a velocity profile. As it has shown in Figure 16, there were two vehicles in the scenario that the front vehicle (the truck) had 60 km/h speed and our vehicle model was 200 meters back of this vehicle with 80 km/h.

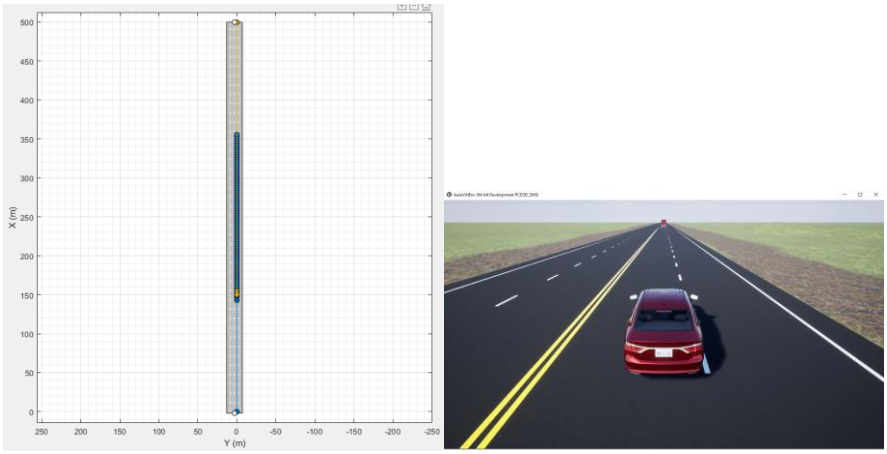

Fig. 16. Our scenario in the drivingScenarioDesigner schematic in MATLAB.

3.4.1.2 Turn

We designed the other scenario for a comparison between our method and the other works. This scenario consists of different turns as shown in Figure 17. The speed limit of this scenario is between 0 to 40 km/h and at the first, the vehicle reaches the 40 km/h with our acceleration limitation that we discussed in constraints.

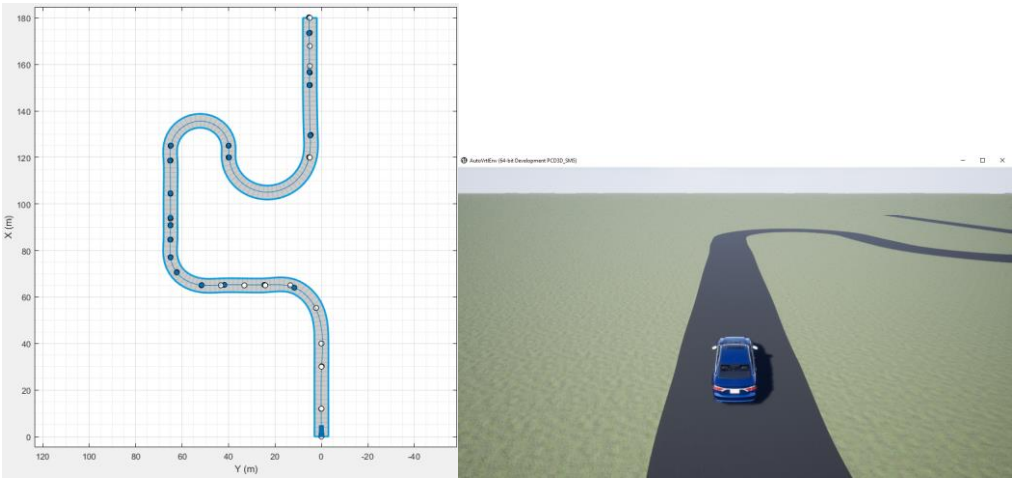

Fig. 17. The Scenario visualization in Unreal Engine.

3.4.2 Adaptive Model Predictive Controller design

We designed our AMPC using mpcDesigner [48] and Simulink. For each time step, our controller updated to make new states for the next prediction horizon. In Simulink, as shown in Figure 18, we used the Adaptive MPC block for this implementation which in it, the constraints and the MPC parameters are attached to it by mpcDesigner tool. The different blocks are to build the requirements of the Adaptive MPC block. We also brought our reference scenarios as discussed before. To determine the prediction horizon and control horizon we did an experimental exercise. The tests led us to define prediction horizon considered as 10 seconds and the control horizon as 5 seconds. The tuning of weights was done by mpcDesigner tuning tool for closed-Loop Performance and State Estimation along with considering the system stability. The constraints, as discussed before, were defined in our controller using the mpcDesigner tuning tool.

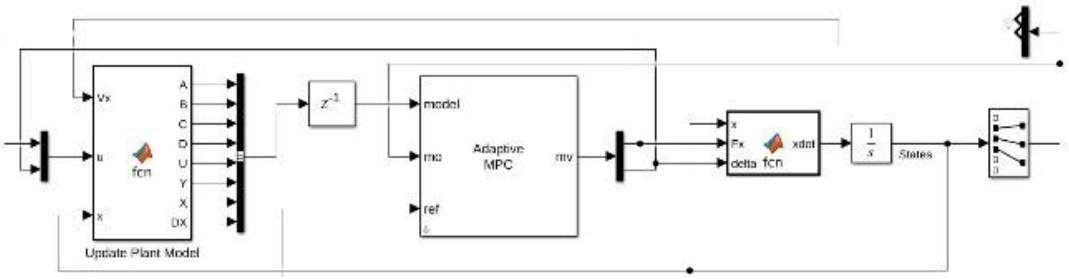

Fig. 18. The Simulink implementation of Adaptive MPC

3.4.3 Simulator

The system is tested with the Unreal Engine simulator [50] which connects to the Simulink. Our simulator considered the scenario data made by drivingScenarioDesigner, and added the output of the system to visualize and evaluate our system.

3.4.4 Embedded platform

The target embedded platform, NVIDIA Jetson AGX Xavier is representative of the next-generation AV Domain Controller. This platform with a GPGPU of 512-core Volta with Tensor Core and a CPU of ARM 8-core v8.2 64-bit is an appropriate choice for the AD systems.

The NVIDIA® Jetson AGX Xavier™ module delivers up to 32 TOPS of accelerated computing capability in a compact form factor consuming under 30 Watts. This gives you more than 20X the performance and 10X the energy efficiency of its predecessor, the NVIDIA Jetson™ TX2.

This advanced system-on-module is powered by the NVIDIA Xavier SoC and designed specifically for autonomous machines. Heterogeneous accelerated computing architecture delivers advanced edge capabilities. Plus, it comes with integrated memory, storage, power management, and an innovative thermal design to enable faster time to market. Run modern AI workloads and solve problems in areas like manufacturing, logistics, retail, service, agriculture, smart cities, and healthcare.

Jetson AGX Xavier is supported by NVIDIA JetPack, which includes a board support package (BSP), Linux OS, NVIDIA CUDA®, cuDNN, and TensorRT™ software libraries for deep learning, computer vision, GPU computing, multimedia processing, and much more. It’s also supported by the NVIDIA DeepStream SDK, which delivers a complete toolkit for real-time situational awareness through intelligent video analytics (IVA). This helps you boost performance and accelerate software development, while reducing development cost and effort.

To have a realistic implementation, we can’t rely on the Matlab/Simulink implementatn, and we utilized embedded coder of MATLAB/Simulink to convert our algorithm into C++ source code, which is then compiled for the target platform.

3.5 Results and discussion

Firstly, we calculated our results regarding of the first scenario, Straight scenario. Then, we investigated the results of the second scenario which is Turn. Finally, we tried to understand our timing results in the embedded platform to be able to use it along with other infrastructures. These plots demonstrate the control system reliability and the correct response. By having a good control system using AMPC we can have the reliable MSDV to evaluate our system.

We evaluated the scenarios by MSDV and IR then we compared our work with the latest works in this area. Our results shown different advantages compared to the previous approaches.

3.5.1 Results of the scenarios

3.5.1.1 Straight road

The straight scenario included two vehicles and a velocity profile. Our vehicle was behind a truck that was slightly far. It started from 0 and reached 80 km/h (22.22 m/s) and as soon as founded the distance of 5 seconds, it started slowing down to maintain the 5 seconds of the distance. Afterwards, it followed the truck by the truck's velocity. As shown in Figure 19, Figure 20, and Figure 21, the output of our controller follows the base-line with a small error.

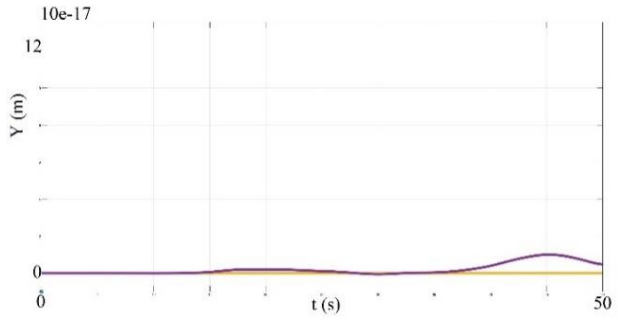

Fig. 19. The plot compares the Y position defined in the scenario (blue) and the result that we achieved in our simulation (orange). As it can be seen the difference in the preiod is almost zero.

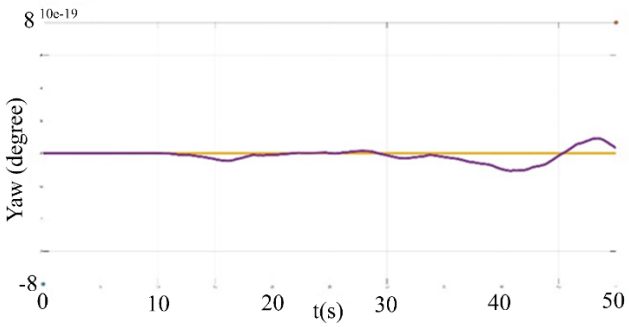

Fig. 20. The plot compares the yaw angle defined in the scenario (blue) and the result that we achieved in our simulation (orange). This shows a high relaibility of the control system.

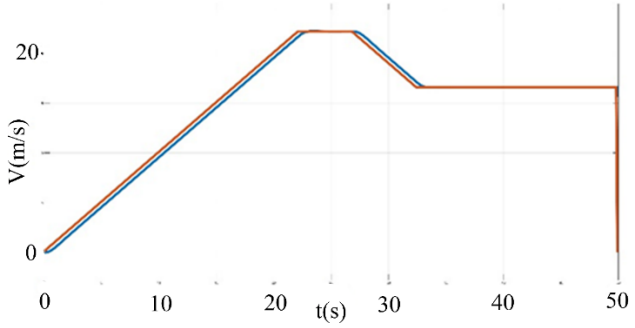

Fig. 21. The plot compares the velocity defined in the scenario (blue) and the result that we achieved in our simulation (orange). The system follows the velocity profile as expected and in the three different changes it adapts itself to the target velocity.

3.5.1.2 Turn

In the turn scenario, we maintained the acceleration limitation based on AMPC algorithm designed by Simulink and mpcDesigner. Figure 22, Figure 23, and Figure 24 show the results.

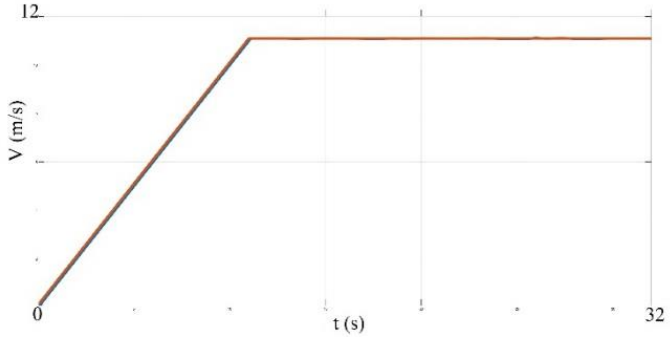

Fig. 22. The plot compares the velocity defined in the scenario (blue) and the result that we achieved in our simulation (orange). The system follows the velocity profile as expected and in the three different changes it adapts itself to the target velocity.

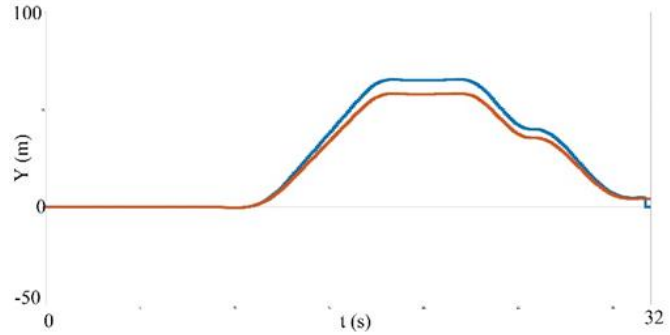

Fig. 23. The plot compares the Y position defined in the scenario (blue) and the result that we achieved in our simulation (orange). As it can be seen the difference in the preiod is almost zero.

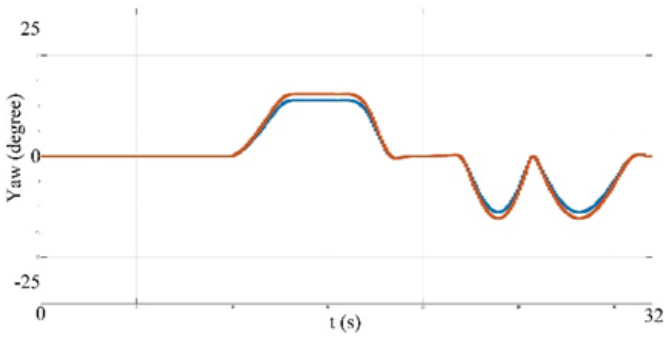

Fig. 24. The plot compares the yaw angle defined in the scenario (blue) and the result that we achieved in our simulation (orange). This shows a high relaibility of the control system.

3.5.2 MSDV and IR analysis

Our evaluation is based on MSDV and IR. IR generally increases overtime during a motion sickening stimulus [51]. In [47], IR is considered as 0 when the passenger feels fine, 1 with a feeling of slightly unwell, 2 as quite ill, and 3 when the passenger is absolutely dreadful. As shown in Table 1, the output of the system more than having a small amount of IR which almost is zero, it has a comparison between the minimum IR of the previous work.

Table 1: The results of the IR evaluation

Scenario	**Time (s)**	**IR (min)**
Straight	50	0.07
Turn	32	0.0017
Turn in [53]	29.73	0.044

Table 1 shows that the IR of the Turn scenario is much lower than the straight one. It is exactly what we expected considering the accelerations used in both scenarios since the Turn scenario has a much lower time of accelerating.

The results show that our performance is better since we try to use the acceleration as small as we can and we try to make it limited to 1 m/s^2. Furthermore, our planner can make an IR near to zero. Therefore, it has a fine feeling according to [47]. It means the comfort criteria is satisfied and the vehicle acceleration/deceleration make the possibility of getting motion sickness lower.

3.5.3 Embedded platform performance

To test the performance of the embedded platform, we ran the system and calculated the timing. When running on the production-like embedded domain controller, our controller achieves 8.7 FPS. It means we can run this motion sickness mitigation system in real-time. This is what exactly it would be needed for the motion sickness mitigation system.

3.6 Conclusion

In this chapter of the thesis, we showed that by having a complex cost function with an emphasis on Motion Sickness Mitigation and consideration of comfort, we can achieve a smooth controller that does not make people sick. This work showed that the AV can have an algorithm for Motion Sickness mitigation along with the other tasks and make the AV more reliable than before.

For the next works, we can add other necessary features of AV such as LiDAR to detect and import the data for the Motion Sickness Mitigation Algorithm. It can finally be an algorithm which is used with the other infrastructures.

We also plan to adopt more complex vehicle models, such as the kinematic and dynamic model, to validate our approach at highest speeds (i.e., > 150km/h), and to possibly include other classes of vehicles, such as busses and coaches, which potentially issue Motion Sickness much more than cars.

Chapter 4

Motion Sickness Minimization Alerting System Using The Next Curvature Topology [41]

Current intelligent car prototypes increasingly move to become autonomous where no driver is required. If an automated vehicle has rearward and forward facing seats and none of the passengers pay attention to the road, they increasingly experience the motion sickness because of the inability of passengers to anticipate the future motion trajectory. In this chapter of the thesis, we focus on anticipatory audio and video cues using pleasant sounds and a Human Machine Interface to display and inform the passengers about the upcoming trajectories that may lead to make the passengers sick. To be able to anticipate the next moves, we require an evaluation system of the next 1 kilometer of the road using the map. The road is investigated based on the amount of the turns and the maximum speed allowed that lead to lateral accelerations that is hig.h enough based on Motion Sickness Dose Value to make the passengers sick. The system alerts the passengers through a Human Machine Interface to focus on the road for prevention of the Motion Sickness. We evaluate our method by using Motion Sickness Dose Value. Based on this work, we can prevent the sickness due to lateral accelerations by making the passengers to focus on the road and decrease the vestibular conflict.

4.1 Introduction

Vehicle control of the semi- and full Autonomous Vehicles should consider the passengers' stress and try to maintain their comfort level [42]. Furthermore, there is a tight relationship between comfort and trust, as well as the automated vehicles' acceptance [43].

One of the wide recognized comfort issues for the passengers probably is Motion Sickness. It starts appearing with headache, pallor, sweating, nausea, vomiting, and disorientation, and they are calculated by Vestibule Ocular Reflex (VOR) parameters, Physiological signals, and Posture stability. To mitigate it, Immersive Experience, Posture and vehicle controllability, and instance visual cues can be used [31].

The potential sources of AV motion sickness can be divided into five groups, namely, loss of controllability and loss of anticipation of motion direction, variation in horizontal and vertical acceleration, Head downward inclination, posture

instability, and lack of synchronization between virtual motion and the vehicle motion profile [31]. The motion sickness is mostly occurred by a conflict between visual and vestibular inputs. However, the loss of controllability over one's movements and unability to predict the movement direction are also crucial in motion sickness [45]. In the most of the cases, the motion sickness experience is for the passengers and the drivers rarely experience it [45]. Possible countermeasures are categorized in two groups: prevention solutions and mitigation solutions.

One of the general ideas for overcoming the motion sickness is using human senses to provide sufficient situation awareness (SA). In the contexts of automotive and driving, SA is recognized as awareness of the current position of the car in relation to its destination, the relative positions, and behavior of other road users and potential hazards, and knowing how these critical variables are likely to change in the near future [74]. This is because the drivers less often get sick since they are able to anticipate the next moves [75] and can predict required actions based on previous experiences.

If a passenger becomes aware of the required information about the road, we can avoid the sensory mismatch. One of the required information is the immediate intention of the AV that involves variation in the lateral and longitudinal forces. This information can be presented shortly before an important situation that is about to happen (for example when a junction is approaching). The virtual modality is one of the ways that the information can be delivered [57]. Furthermore, the informative auditory lowers the average illness ratings respect to the condition without informative cues [56].

We contribute to research with the original design of a minimization system that predicts the road characteristics in one kilometer ahead and using the HMI instructions for the passengers to be ready for the next potential motion sickness. The system calculates in real time using an NVidia AGX and monitors the road all the time. The system is designed in a way that prevents the unnecessary interactions with the passengers both visual and sound cues and the system is on a real vehicle with a motion prediction algorithm to describe the next moves. Despite most of the works that are in a simulation phase, our work is tested in real vehicles in real scenarios. To evaluate our work, we use Motion Sickness Dose Value (MSDV) [47] for the evaluation. Our contributes can be categorize in the following groups:

- A real time system that calculates the potential lateral accelerations based on the road characteristics in the next 1 km;

- An alert system that tries to interact with the passengers only at the time of the existing the potential motion sickness ahead and tries to minimize the interactions;

- Defining a new equation to calculate the motion sickness in the curves;

- The system can be used in fully autonomous vehicles as well as vehicles with less autonomy.

In the following sections, we first review the state-of-the-art in motion sickness. Then we describe the details of our Human Machine Interface (HMI) and sound profile. Finally, we show how we implemented it and discuss the experimental results with respect to the reference metrics of motion sickness.

4.2 Curvature and lateral analysis

For having a correct lateral acceleration prediction, we need to the speed along with the maximum superelevation rate and the maximum allowable side friction demand (assumed in the Green Book [76] to be the friction between the tires and pavement) determine the minimum radius of curvature for each design speed [77]. This is necessary to go through these equations and describe them since we plan to use the final output in the MSDV equation to retrieve the final formula. Equation 1 is used to determine the minimum radius of a circular horizontal curve.

$$R_{min} = \frac{V_d^2}{15(e_{max} + f_{max})} \quad (1)$$

where,

R_{min} = minimum radius of curvature (ft),

V_d= design speed (mph),

e_{max} = specified maximum superelevation elevation rate (fit/100 ft),

f_{max} = specified maximum side friction demand.

The tendency of a vehicle to either skid off the road or overturn must be resisted by either the friction developed between the vehicle tires and the pavement or the vehicle's roll stability, respectively. A vehicle will skid off the road when the side

friction demand exceeds tire/pavement friction. Also, a vehicle will overturn if the unbalanced lateral acceleration exceeds the rollover threshold of the vehicle.

According to the Green Book [76], the maximum available side friction factor (friction developed by the tire-pavement interaction) should not be used directly for the design of a horizontal curve. Instead, the value used in design should be a percentage of the maximum available side friction factor that can be used with comfort and safety by the majority of drivers. This limiting value is described as the lateral acceleration that is sufficient to cause the driver to experience discomfort and to instinctively avoid higher speed. Accordingly, the speed at which a driver feels discomfort due to the lateral acceleration generated while traversing a curve can be accepted as a design control for the maximum allowable amount of the side friction factor.

The Green Book [76] provides side friction factors (J) for low-speed and high-speed design of roadways. Currently, American Association of State Highway and Transportation Officials (AASHTO) bases these recommended values on the results of various. AASHTO's recommended maximum allowable side friction factors for low-speed roads vary with the design speed from 0.38 at 10 mph (16 km/h) to 0.14 at 45 mph (72 km/h), and then vary directly with the design speed to 0.08 at 80 mph (128 km/hr). These values for high speed provide a "reasonable margin of safety at high speeds." The values for low-speed design are higher since drivers are more tolerant of discomfort at lower speeds.

The coefficients of friction for forward skid on wet concrete pavement with tires having new treads. The wet pavements have lower coefficients of friction than dry pavement. Currently, the Green Book defines the margin of safety in horizontal curve design as the difference between f_design and fat impending skid. These values for at impending skid are assumed to be the ultimate side friction values for good tires on wet concrete pavement. These conditions are considered sufficiently representative for a meaningful analysis.

4.2.1 The Point Mass Model

Under the AASHTO policy, a point mass is used to represent a vehicle on a horizontal curve. In this model, the vehicle’s suspension is ignored. From basic physics, the lateral acceleration of a point mass traveling on a circular path at a constant speed can be represented by the following relationship:

$$a = \frac{V^2}{15R} \qquad (2)$$

where,

a = lateral acceleration (g)

V = vehicle speed (mph)

R = radius of curve (ft)

The lateral acceleration experienced by the vehicle is relative to g which is equal to 32.2 ft/s^2 (9.8 m/s^2).

In the Point Mass Model, all points in a vehicle are assumed to have the same acceleration; in other words, the entire vehicle is a "point mass." Consider in Figure 25.

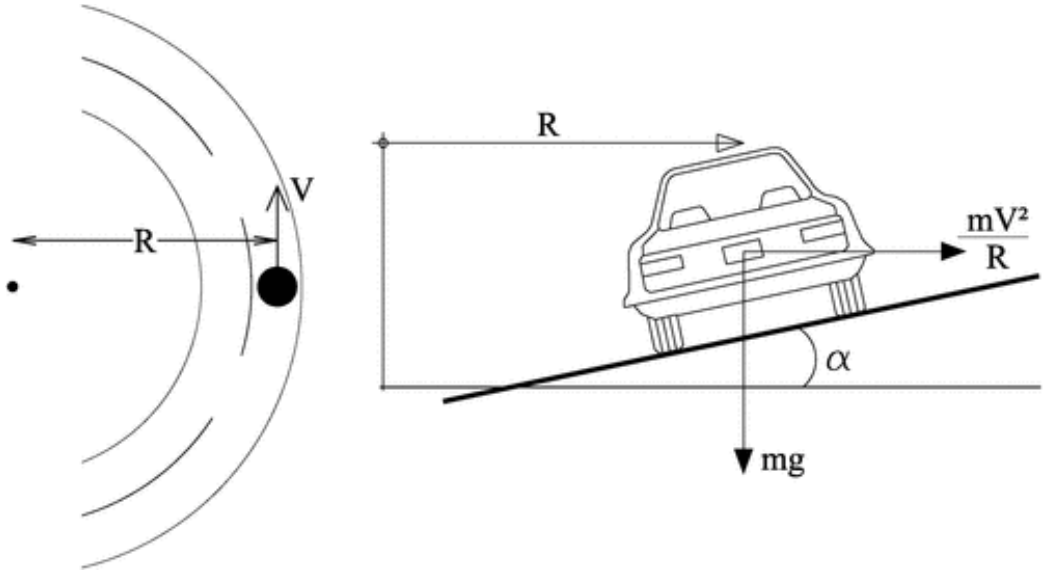

Fig. 25. The mass point of the vehicle and the radius (R) of the curve.

where a vehicle is represented as a point with mass, *m*, and weight, *mg*, traversing at speed, *V*, around a curve with radius, *R*, and superelevation, *e*. Summing forces along the superelevated plane results in the following equation:

$$fN + mgsin\theta = \frac{mV^2}{R}cos\theta \qquad (3)$$

where,

f = side friction demand,

N = normal force resulting from force of vehicle due to gravity, mg,

W = force of vehicle due to gravity, mg,

g = acceleration due to gravity, 32.3 fit/s^2 (9.81 m/s^2),

θ = angle resulting from superelevation, e.

Respect to this formula, f equals to:

$$f = \frac{\frac{mV^2}{R}cos\theta - mgsin\theta}{N} \quad (4)$$

With small angle approximation where $cos\theta$ is 1 and $sin\theta$ is e, and based on the description provided in [103], results in:

$$f = \frac{V^2}{15R} - e \quad (5)$$

4.2.2 Lateral acceleration

When a vehicle moves in a circular path, it undergoes a centripetal acceleration that acts toward the center of curvature. This acceleration is sustained by a component of the vehicle's weight related to the roadway superelevation, by the side friction developed between the vehicle's tires and the pavement surface, or by a combination of the two. Centripetal acceleration is sometimes equated to centrifugal force. However, this is an imaginary force that motorists believe is pushing them outward while cornering when, in fact, they are truly feeling the vehicle being accelerated in an inward direction. In horizontal curve design, "lateral acceleration" is equivalent to "centripetal acceleration"; the term "lateral acceleration" is used in this policy as it is specifically applicable to geometric design.

Based on [78], large radius curves, the drivers limit their speed by both their comfortable lateral acceleration and speed environment. On small curves, a comfortable or "easy ride" corresponded to an experienced lateral acceleration of 0.35g to 0.40g.

Based on the radius and maximum velocity defined in the road, we may find the actual acceleration that will be occurred in the curve and calculate the MSDV based on it.

4.2.3 Radius calculation

As discussed, for having the lateral acceleration in vehicle, we need the radius of the curve. To calculate the radius, we use the pure pursuit method. To use it we need to choose a proper look ahead distance, based on the Figure 26.

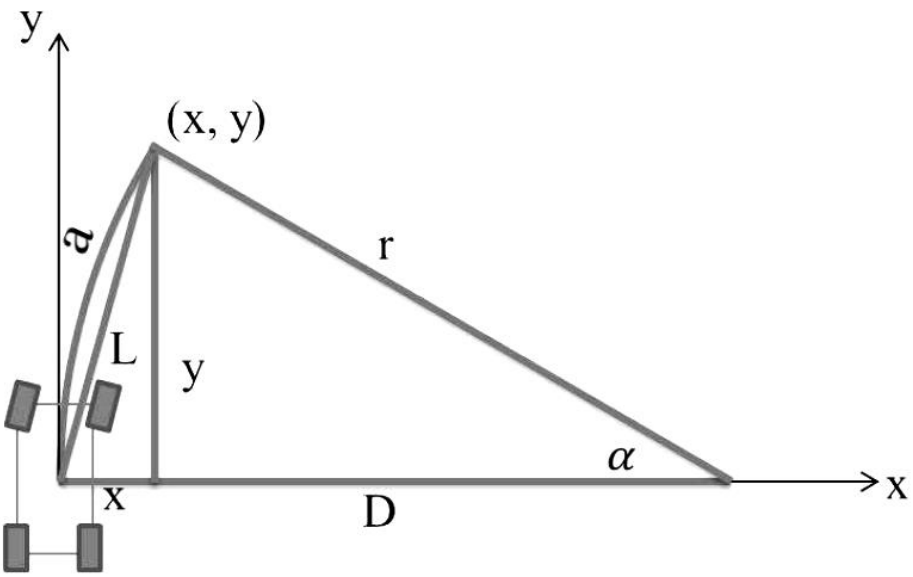

Fig. 26. Pure pursuit model geometry. This model is used to calculate the radius of the curvature. Based on the curvature's radius along with the velocity and acceleration in the curvature, we will be able to calculate the MSDV and IR.

The x and y axis construct the coordinate system of machine. The point (x, y) is a point some distance ahead of the machine. The L is the length of the cord of the arc connecting the origin to the point *(x, y). r* is the radius of curvature of the arc and a is the arc length of α angle. The relationship of *x, L* and *r* (the same as *R* in the previous section) is as follows:

$$D + x = r \quad (6)$$

$$D^2 + x^2 = r \quad (7)$$

$$x^2 + y^2 = L^2 \quad (8)$$

From Eq. (6), Eq. (7) and Eq. (8),

$$r^2 - 2rx + x^2 + y^2 = r^2 \quad (9)$$

$$r = \frac{L^2}{2x} \quad (10)$$

$$a = \frac{\alpha}{360}(2\pi r) \quad (11)$$

By choosing a look-ahead distance and calculating the path error x, the radius of the curvature required to get the machine on the required path can be calculated.

4.2.4 Motion Sickness Dose Value

The total MSDV resulted from lateral and longitudinal motion is given as [47]:

$$MSDV = \sqrt[2]{\int_0^T (a_{x,w}\,(t))^2} + \sqrt[2]{\int_0^T (a_{y,w}\,(t))^2} \quad (12)$$

Where $a_{y,w}(t)$ and $a_{x,w}(t)$ are the frequency weight acceleration in the lateral and longitudinal direction.

Where $a_{y,w}(t)$ and $a_{x,w}(t)$ are the frequency weight acceleration in the lateral and longitudinal direction.

$$a_{x,w}(t) = a_x(t) \times W_f \quad (13)$$

$$a_{y,w}(t) = a_y(t) \times W_f \quad (14)$$

where $a_x(t)$ is the longitudinal acceleration and $a_y(t)$ is the lateral one. In Standard 6841 [47] W_f is defined as the weighting factor for evaluating low frequency motion with respect to motion sickness. Since we consider only the lateral accelerations, we consider just $a_y(t)$ in our calculations. From the standards [47], [73], a simple linear approximation between mean passenger illness rating and MSDV is defined as:

$$IR = K \times MSDV \quad (15)$$

where IR is defined as the predicted illness rating and K is an empirically derived constant. Based on [47] and [73], the illness rating value is in four levels; The illness rating of 0 demonstrates the feeling fine, 1 demonstrates slightly unwell, 2 demonstrates quite ill, and 3 demonstrates absolutely dreadful.

4.2.5 Motion Sickness in the curves

The previous calculations show that we can calculate the MSDV using the lateral accelerations and the lateral accelerations can be defined based on the velocity and the radius of the curve. To achieve a single formula, we neglect the accelerations in x axis since we assume that we will have constant velocity in the curves. Therefore, MSDV will be:

$$MSDV = \sqrt[2]{\int_0^T (a_y(t) \times W_f)^2} \quad (16)$$

Since we considered a constant velocity on the curve, our acceleration will not change in the curve and based on the Equation (2), the Equation (16) we will have:

$$MSDV = \frac{V^2}{15R} \times W_f \sqrt[2]{T} \quad (17)$$

With this new MSDV equation that we have defined, we can calculate the MSDV before each curve. By calculating each MSDV before the curve, we will be able to decide whether it would be a road with potential motion sickness or not.

4.3 The experimental setup

For the experimental setup of our work, we used a simulator sending the data constantly to our Human Machine Interface (HMI) and embedded system to communicate with the passengers. The embedded platform has the responsibility of calculating the potential MSDV based on the next lateral acceleration and alert the passengers through the HMI about the upcoming condition that may lead to Motion Sickness. Figure 27 demonstrates the diagram of our system.

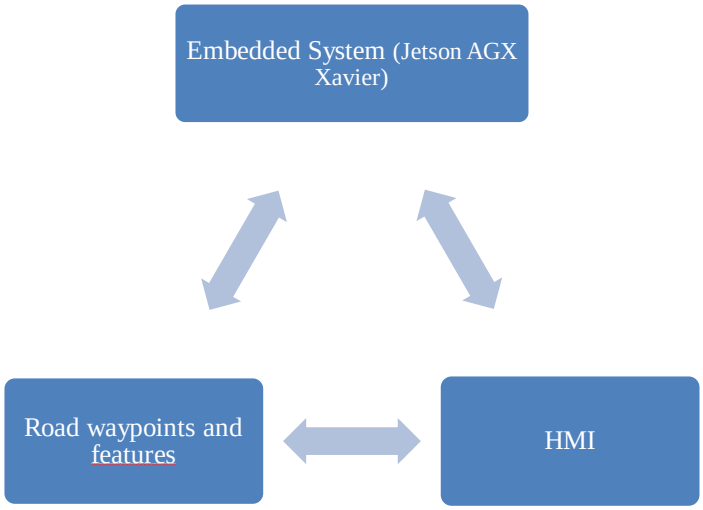

Fig. 27. The diagram of the Motion Sickness minimization system.

4.3.1 Embedded system

We targeted NVIDIA Jetson AGX Xavier that is representative of the next-generation AV Domain Controller as our embedded platform. This embedded platform has a GPGPU of 512-core Volta along with Tensor Core and a CPU of ARM 8-core v8.2 64-bit and would be a suitable choice for our system.

4.3.2 Human Machine Interface

We used a Human Machine Interface (HMI) to interact with the passenger. We do this on a window with message alerting the passenger about starting to focus on the upcoming road. In this way the passengers know about the potential upcoming motion sickness and try to concentrate on the road to minimize it. Figure 28 shows the HMI we used to interact with the passengers.

Fig. 28. The HMI that interacts with the passengers. In this HMI we try to alert the passengers about the upcoming roads with potential motion sickness.

4.3.3 Road waypoints and features

For the testing of the work, we used the ego_pose of nuScences dataset [79]. The nuScenes dataset is the first dataset to carry the full autonomous vehicle sensor suite: 6 cameras, 5 radars and 1 lidar, all with full 360-degree field of view. nuScenes comprises 1000 scenes, each 20s long and fully annotated with 3D bounding boxes for 23 classes and 8 attributes. We used the ego pose of the dataset and gathered all the necessary information to test our work. The ego_pose has been extracted by the MATLAB drivingScenario tool. Then we created the waypoints by its poses. The features that should be received by the embedded system are the width and the waypoints of the center line of the road. The waypoints should include *x*, *y*, and *z* dimensions. Based on this information and the theory that we mentioned before, we calculate the potential lateral acceleration and MSDV.

4.4 Tests and results

Testing of our work was done by the nuScenes dataset and exporting the MSDV into the simulator. In MATLAB we extracted the results of the MSDV and exported

them to the simulator as explained in Figure 29. If the illness rating is more than 1, any symptoms, however slight [73], the HMI would show the MSDV alert.

Fig. 29. The test procedure that starts with the getting the ego_pose of the nuScenes dataset. The ego_pose would be imported to Matlab and create the scenario by the driving Scenario tool. Then, the MSDV would be calculated in the Embedded system and sends the results to the HMI to show if the Motion Sickness is coming or no.

4.4.1 The road testing

The tests utilized the data acquired from the road dataset. The Figure 30. shows one of the tests that have been conducted through a real waypoint from dataset. In the Figure 30. can be seen a curvature that has been distinguished as a potential curve of motion sickness.

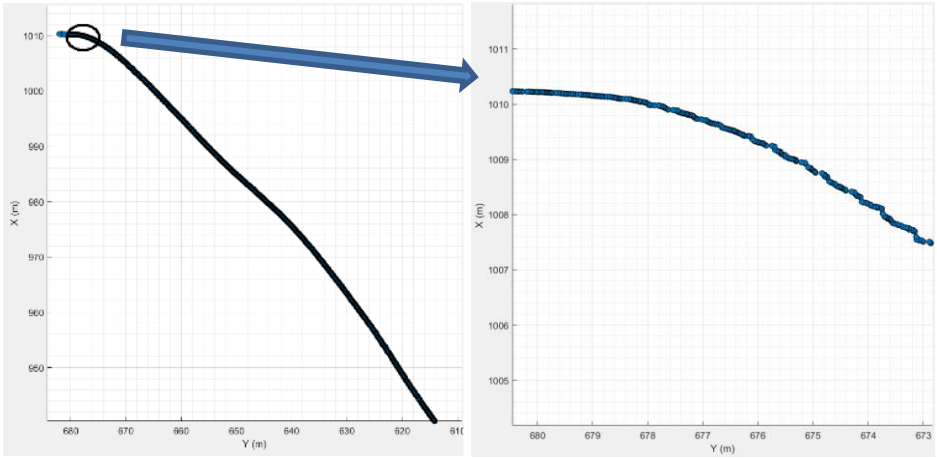

Fig. 30. The potential curve of motion sickness. Each curvature that is a potential curve for the motion sickness, is investigated by the possible MSDV and IR based on the maximum velocity and the curvature's radius.

4.5 Conclusion

In this chapter of the thesis, we demonstrated a novel way to alert the passengers for the upcoming motion sickness. This system aims the compatibility for using in fully or semi-autonomous vehicles. The new equation of the motion sickness made us enable to calculate the level of the motion sickness by the lateral acceleration for

the next curvatures. The alerting system can help the passengers to prevent the motion sickness.

This work by its functionalities enables us to extend it in a real world. For the future improvements, we plan to use it in the real vehicles by the online mapping. The online map services, like google maps, would help us to use the ahead positions and with those positions and our equations we will be able to calculate the MSDV and alert the passengers online.

Chapter 5

Motion Prediction using Attention Heads and Traffic rules in intersections

In the past two chapters we introduced our novel methods for motion sickness mitigation. As declared before, one of the sources of the motion sickness is the ability to anticipate the direction of movement. It means that if we want to improve the ability of anticipation the movement direction, we need to have a correct methodology to do that. Therefore, the motion sickness mitigation led us to work on the motion prediction methodologies. In this regard, we tried to investigate the state-of-the-art motion prediction methods and implement some of the most important ones. This work opened a new are of working on motion prediction and a proposal of a new model to be compared with the state-of-the-art methods. In the other hand, autonomous driving motion forecasting is essential to have a correct and reliable planning. The influence of the road agents on each other makes it even more challenging. However, most prior works have not considered these interactions and planning against fixed predictions would reduce the ability to represent the future interaction possibilities between different agents. In this chapter of the thesis, we propose a model that predicts the agents' behavior in a jointly manner. We take advantage of using masking strategy as the query to our model. Our model architecture uses a unified Transformer architecture by employing attention across the road elements, agent interactions and traffic rules in intersections. We evaluate our approach on autonomous driving datasets for behavior prediction and test it on Carla simulator. Our work demonstrates that motion forecasting by a model with a masking strategy and having attentions and traffic rules can lead us to a state-of-the-art model. The result of our work is compared with the state-of-the-art models from the leaderboard of Argoverse and nuScenes.

5.1 Introduction

Predicting the behavior of the road agents involves different factors and their behaviors of other agents may affect dramatically on the planning. A prerequisite

of such a multi-task system is that it needs to be able to jointly predict the futures of multiple agents (including the autonomous vehicle), while simultaneously taking into account their interactions. The interaction prediction tasks require that models predict the joint futures of multiple agents, and the models are expected to produce future predictions for all agents such that the agents' futures are consistent within each future. A simple variant of self-attention [80] is employed in which the attention mechanism is efficiently factorized across the agent-time axes. The resulting architecture simply alternates attention between dimensions representing time and agents across the scene, resulting in a computationally efficient, uniform, and scalable architecture.

In the other hand, High-Definition maps (HD-maps) provide extremely useful geometric and semantic information for motion forecasting, as the behaviours of actors largely depend on the map topology. For example, a vehicle is unlikely to take a left turn when there is not a left turn lane nearby. Effectively exploiting HD maps is essential for motion forecasting models to produce plausible and accurate trajectories.

First attempts exploit HD maps as heuristics. Actors are first associated with lanes and all candidate motion paths are then generated based on map topology. In this way, the prediction results are constrained by the map. However, this approach cannot capture rare and non-compliant behaviours, which while not very likely, might be safety critical.

Recent works use machine learning to learn semantic representations from maps. To enable HD maps to be processed by neural networks the map data is rasterized to create image-like raster inputs. Map topology is implicitly encoded as lines, masks or colours, which are then processed by a 2D Convolutional Neural Network (CNN). These learned map features were shown to provide useful context information for motion forecasting. However, these approaches have two disadvantages. First, the rasterization process inevitably results in information loss. Second, maps have a graph structure with complex topology which 2D convolution may be very inefficient to capture. For example, a lane of interest may extend for a long range in the lane direction. To capture this information, the receptive field has to be very large, covering not only the intended area, but also large areas outside the lane. Furthermore, lane pairs in the same or opposite directions have completely different semantic meanings and dependencies, although the lanes in both pairs are spatially close to each other.

In a routable network, intersections are essential junctions that connect different roadways. Most of the algorithms treat intersections as points at which roadways

are connected, and the connectivity that guides approaching vehicles is not specified explicitly. However, an intersection represents a compact junction for which a set of turning paths and forbidden turning times are fundamental for the schedule of approaching vehicles. Even to adapt to changes in urban traffic flow, the layout of turning paths and traffic rules at intersections may be adjusted dynamically, i.e., left turn banned from during rush hour. A comprehensive approach focusing on the detection of both locations and turning paths, including the forbidden turning time, of intersections using a large number of vehicle GPS trajectories is used.

An intersection is a road junction where two or more roadways either meet or cross. Various road markings, traffic lights and traffic signs schedule the approaches of vehicles to the intersection at appropriate speeds and prevent vehicle crashes.

Our main contributions In this chapter of the thesis are:

- Developing the previous works on Lane Graph Convolutional Network by adding the Traffic Rules as an attention mechanism,
- Using a high demand road junctions and complex intersections, as the point of adding the Traffic Rules into the model,
- A Transformer-based architecture factored over agents, time, and road graph elements that exploits the inherent dependencies of the problem and the traffic rules,
- Using different datasets to evaluate our work such as nuScenes and Argoverse.
- Achieving the state-of-the-art results comparing with the other works that their model works in the intersection areas.

5.2 Motion forecasting in AV literature

A common approach for short-term prediction of future motion is to assume that the driver will not change any control inputs (such as steering and acceleration) using techniques such as a Kalman filter (KF). This approach first associates detected vehicles with one or more lanes from the map. Then, all possible paths are generated for each (vehicle, associated lane) pair based on map topology, lane connectivity, and vehicle's state. Classical machine learning approaches such as Hidden Markov Model, Bayesian networks or Gaussian Processes have been

applied to motion prediction in autonomous driving. However, these methods require manually designed features and no longer provide state-of-the-art performance. Most recent research on motion prediction employs deep networks. In one line of research, recurrent neural networks (RNN) with Long Short-Term Memory (LSTM) or gated recurrent unit (GRU) were applied to predict future trajectories from past observed positions. Going beyond just using past observed positions as inputs, rasterize actor's surrounding context and other scene information in a bird's-eye view (BEV) image.

Conventionally, three types of approaches exist for vehicle trajectory prediction. Physics-based, maneuver-based, and interaction-aware. Physics-based methods usually consider vehicle kinematic and dynamic constraints, such as yaw rate and acceleration rate, and environmental factors, such as the friction coefficient of a road surface. They assume that the vehicle's motion depends only on physical equations of motion. They are the simplest models (e.g., constant velocity, constant acceleration) with low computational complexity and, as a result, their predictions are typically only reliable for a short horizon. This approach can achieve short-term predictions (<1 s). Maneuver-based motion models assume that the vehicle's motion can be represented by a series of maneuvers executed independently of other vehicles. Maneuver-based approaches, the future trajectory of the target is predicted by identifying the maneuver in execution from a finite set of maneuvers contained in a database. Methods to identify the maneuver include Hidden Markov Models (HMM) [104] and Gaussian Processes (GP) [105]. Typically, these approaches also fail to consider the interactions between vehicles. Most physics-based and maneuver-based approaches do not account for interactions among vehicles. This has motivated the development of interaction-aware methods that take into account the interdependencies of vehicle maneuvers for trajectory prediction. Attention mechanisms can be naturally integrated with RNN to improve the model explainability.

Algorithms for generating intersections from vehicle trajectories can be grouped into two basic types. One type directly extracts intersections using either the geometric characteristics of trajectories at intersections or the spatial relationships between multiple trajectories at intersections. For example, [81] identified intersections for the first time using an advanced shape descriptor to analyse the specific patterns of the heading changes in tracking points. [82] obtained the locations of crossings by intersecting two pedestrian trips. [83] detected intersections by analysing the densities of large-angle intersection points among neighbouring vehicle trajectories. [84] characterized the intersection features of an

urban transportation network with good connectivity, a high density of trajectories and multiple traversing trajectory patterns to detect intersection locations.

The typical incremental method for generating road maps developed by [85] added a single trajectory to a graph by considering the relationship between the input points from the trajectory and the existing graph and by determining whether a new node and edge must be created. Clustering trajectories to generate road centrelines is another important approach [86-89].

5.3 Implementation of the State-Of-The-Art works

After consideration the literature, we concluded to use some previous state-of-the-art works to implement. In this way, we trained our model in the nuScenes [79] and Argoverse [90] datasets (description in section 5.4.2.3). First, we implemented Trajectron++ [91] and then, LaneGCN [92].

5.3.1 Trajectron++

Trajectron++ is an open and extensible approach built upon the Trajectron [93] framework which produces dynamically feasible trajectory forecasts from heterogeneous input data for multiple interacting agents of distinct semantic types. The Trajectron++ key contributions are twofold: First, they show how to effectively incorporate high-dimensional data through the lens of encoding semantic maps. Second, they propose a general method of incorporating dynamics constraints into learning-based methods for multi-agent trajectory forecasting. Trajectron++ is designed to be tightly integrated with downstream robotic modules, with the ability to produce trajectories that are optionally conditioned on future ego-agent motion plans. They present experimental results on a variety of datasets, which collectively demonstrate that Trajectron++ outperforms an extensive selection of state-of-the-art deterministic and generative trajectory prediction methods, in some cases achieving 60% lower average prediction error. At a high level, a spatiotemporal graph representation of the scene in question is created from its topology. Then, a similarly-structured deep learning architecture is generated that forecasts the evolution of node attributes, producing agent trajectories as shown in Figure 31a. Figure 31b shows our implementation in Carla simulator.

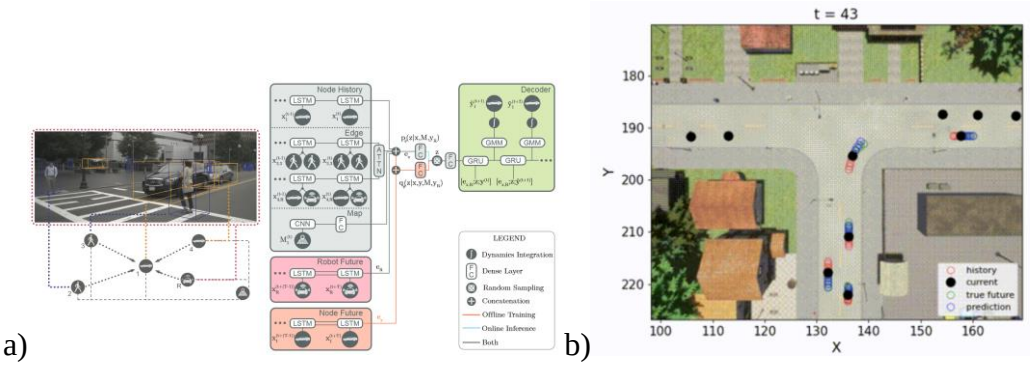

Fig. 31. a) The trajectron++ architecture and b) its implementation in Carla simulator.

As shown in Figure 31a, their approach represents a scene as a directed spatiotemporal graph. Nodes and edges represent agents and their interactions, respectively. Each note has its corresponding network architecture and all of these nodes have interactions based on the edges that are involved.

5.3.2 LaneGCN

They construct a lane graph from raw map data and use LaneGCN to extract map features. In parallel, ActorNet extracts actor features from observed past trajectories. Then FusionNet [94] is used to model the Interactions between actors themselves and the map and predict the future trajectories. Figure 32 shows our implementation in Carla simulator.

In their model ActorNet receives the past actor trajectories as input, and uses 1D convolution to extract actor node features and MapNet constructs a lane graph from HD maps, and uses a LaneGCN to exact lane node features. Then FusionNet is a stack of 4 interaction blocks. The actor to lane block fuses real-time traffic information from actor nodes to lane nodes. The prediction header uses after-fusion actor features to produce multi-modal trajectories.

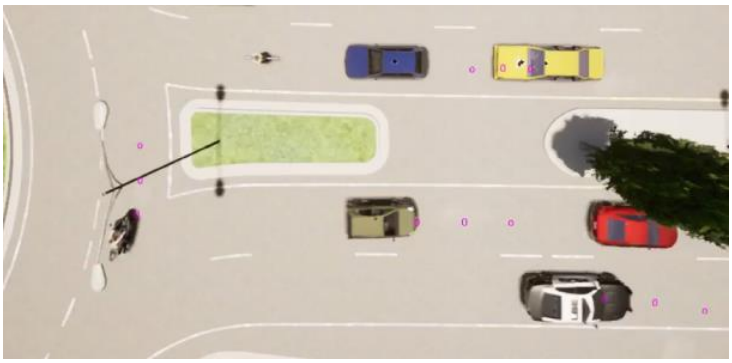

Fig. 32. The LaneGCN implementation in Carla simulator

5.4 Motion Prediction Using Attention Heads and Traffic rules in Intersection

5.4.1 Architecture

To find the best trajectories ahead, we proposed a novel method to create a model using three most important modules. The three modules as shown in the Figure 33 are Actor Features, Map Features, and Traffic Rules.

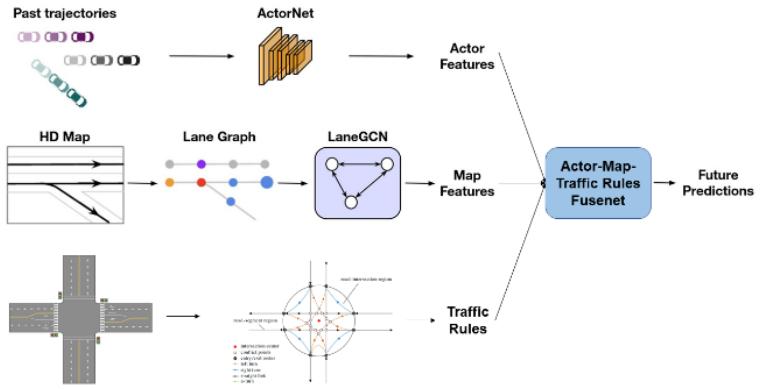

Fig. 33. The topology of our method to develop the model using Trajectories, HD Maps, and the Traffic rules in the intersections.

5.4.1.1 Actor Features

It is assumed that actor data is composed of the observed past trajectories of all actors in the scene. Each trajectory is represented as a sequence of displacements $\{\Delta p_{-(T-1)}, \ldots, \Delta p_{-1}, \Delta p_0\}$, where Δp_t is the 2D displacement from time step $t-1$ to t, and T is the trajectory size. All coordinates are defined in the Bird's Eye View (BEV), as this is the space of interest for traffic agents. For trajectories with sizes smaller than T, padding them with zeros. Adding a binary $1 \times T$ mask to indicate if the element at each step is padded or not and concatenate it with the trajectory tensor, resulting in an input tensor of size $3 \times T$.

While both CNNs and RNNs can be used for temporal data, here an 1D CNN is used to process the trajectory input for its effectiveness in extracting multi-scale features and efficiency in parallel computing. The output of ActorNet is a temporal feature map, whose element at $t = 0$ is used as the actor feature. The network has 3 groups/scales of 1D convolutions. Each group consists of 2 residual blocks, with the stride of the first block as 2. A Feature Pyramid Network (FPN) [95](Lin et al., 2017) is used to fuse the multi-scale features and apply another residual block to obtain the output tensor. For all layers, the convolution kernel size is 3 and the

number of output channels is 128. Layer normalization [96] and the Rectified Linear Unit (ReLU) [97] are used after each convolution.

5.4.1.2 Map Features

A novel deep model, called MapNet, to learn structured map representations from vectorized map data is used. This contrasts previous approaches, which encode the map as a raster image and apply 2D convolutions to extract features.

Map Data: In this chapter of the thesis, it is adopted a simple form of vectorized map data as our representation of HD maps. Specifically, the map data is represented as a set of lanes and their connectivity. Each lane contains a centerline, i.e., a sequence of 2D BEV points, which are arranged following the lane direction (see Figure 33, top). For any two lanes which are directly reachable, 4 types of connections are given: predecessor, successor, left neighbor and right neighbor. Given a lane A, its predecessor and successor are the lanes which can directly travel to A and from A respectively. Left and right neighbors refer to the lanes which can be directly reached without violating traffic rules. This simple map format provides essential geometric and semantic information for motion forecasting, as vehicles generally plan their routes by reference to lane centerlines and their connectivity.

Lane Graph Construction: Instead of encoding maps as raster images, it is derived a lane graph from the map data as the input. In designing the lane graph, it is expected of its nodes to have a fine resolution. Given any actor location, query the lane graph and find its nearest nodes to retrieve accurate map information is done. From this point of view, it is not an optimal choice to directly use the lane centerlines as the nodes.

Referred to Figure 34 for an abstraction of the lane graph construction. It is first defined a lane node as the straight-line segment formed by any two consecutive points (grey circles in Figure 34) of the centerline. The location of a lane node is the averaged coordinates of its two end points. Following the connections between lane centerlines, it is also derived 4 connectivity types for the lane nodes, i.e., predecessor, successor, left neighbour and right neighbour. For any lane node A, its predecessor and successor are defined as the neighbouring lane nodes that can travel to A or from A respectively. Note that one can reach the first lane node of a lane lA from the last lane node of lane lB if lB is the predecessor of lA. Left and right neighbours are defined as the spatially closest lane node measured by `2 distance on the left and on the right neighbouring lane respectively. It is denoted the lane nodes with V ∈ RN ×2, where N is the number of lane nodes and the i-th row of V

is the BEV coordinates of the i-th node. It is represented the connectivity with 4 adjacency matrices {Ai}i∈{pre,suc,left,right}, with Ai ∈ RN ×N . It is denoted Ai,jk, as the element in the j-th row and k-th column of Ai. Then Ai,jk = 1 if node k is an i-type neighbor of node j.

LaneConv Operator [92]: A natural operator to handle lane graphs is the graph convolution. The most widely used graph convolution operator is defined as Y = LXW , where X ∈ RN ×F is the node feature, W ∈ RF ×O is the weight matrix, and Y ∈ RN ×O is the output. The graph Laplacian matrix L ∈ RN ×N takes the form L = D−1/2(I +A)D−1/2, where I, A and D are the identity, adjacency and degree matrices respectively. I and A account for self-connection and connections between different nodes. All connections share the same weight W, and the degree matrix D is used to normalize the output. However, this vanilla graph convolution is inefficient in our case due to the following reasons. First, it is not clear what kind of node feature will preserve the information in the lane graphs. Second, a single graph Laplacian cannot capture the connection type, i.e., losing the directional information carried by the connection type. Third, it is not straightforward to handle long range dependencies, e.g., akin dilated convolution, within this form of graph convolution.

Node Feature: First, it is required to define the input feature of the lane nodes. Each lane node corresponds to a straight-line segment of a centerline. To encode all the lane node information, it is needed to take into account both the shape (size and orientation) and the location (the coordinates of the center) of the corresponding line segment.

5.4.1.3 Traffic Rules

The prediction would be even more precise if we consider some traffic rules. A traffic rule can change the prediction based on the conditions of the Agents specially in the intersections. The approach would be the predictions in the intersections to where a complex agent set are interacting with each other, and the traffic rules can impact on the trajectories.

An intersection is a road junction where two or more roadways either meet or cross. Various road markings, traffic lights and traffic signs schedule the approaches of vehicles to the intersection at appropriate speeds and prevent vehicle crashes.

Figure 34 is an intersection diagram to allow for a visual understanding of our approach. To provide clear navigation information for routing, the intersection is represented as a network graph (Figure 34). In such a model, the location of an intersection and its traffic rules can be easily modelled by considering the spatiotemporal characteristics of the vehicle GPS trajectories near the intersection using the following scheme.

Considering that intersections are strongly correlated to the curved parts of vehicle GPS trajectories (henceforth referred to as turns), it can be assumed that turns generally occur at intersections rather than on roadways. Based on this assumption, a comprehensive approach for detecting intersections and extracting traffic rules was developed, where three essential steps are illustrated. First, a density grid of turns is generated. The value of a cell represents the number of turns passing over the cell. Next, the locations and extensions of the intersections are detected using density analysis. Finally, the traffic rules of the intersections are determined by clustering the time series dataset of the tracking points. In the Figure 34 an intersection is shown with its possible drivable paths.

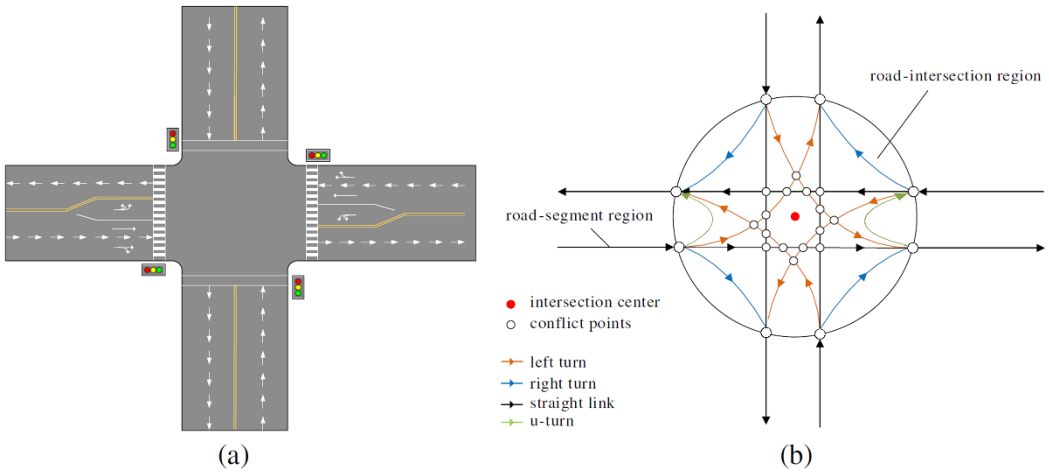

Fig. 34. The intersection topology.

After consideration the intersection topology, we need to define the way of the detection of the intersections in the roads and including the traffic rules in our method. As discussed, we trained our model in nuScenes [79] and Argoverse[90] datastes. Considering the nuScenes, the map database consists of multiple layers where each layer is made up of records. Each record will have a token identifier. In this case, we used the traffic light token, a physical world's traffic light. This layer has some attributes like *traffic_light_type* that denotes whether the traffic light is

oriented horizontally or vertically, *from_road_block_tokens* that denotes from which road block the traffic light guides, *items* that are the bulbs for that traffic light, and *pose* that denotes the pose of the traffic light. We can also get the drivable area of the lanes. Collecting all these information, we build a network as shown in Figure 34.

5.4.2 Implementation and results

In this section, we describe how the implementation was done. We have used python with machine learning libraries for training and verifying our model and then we tested our model using the Carla simulator. In python we used the machine learning libraries and we used the nuScenes and Argoverse datasets to develop our model. Then the model has been tested using Xavier AGX as the embedded platform.

5.4.2.1 Python implementation

We have used Machine Learning libraries (Pytorch and Tensorflow) in python to develop our model. In this case, we were able to access the datasets to train the model and test it in the test datasets.

Machine learning is a field in computer science where existing data are used to predict, or respond to, future data. It is closely related to the fields of pattern recognition, computational statistics, and artificial intelligence. Machine learning is important in areas like facial recognition, spam filtering, and others where it is not feasible, or even possible, to write algorithms to perform a task.

To describe what we have implemented, it is necessary to introduce the elements of Machine learning that can be useful in our implementation. These components will be discussed as bellow.

Data

All learning methods are data driven. Sets of data are used to train the system. These sets may be collected by humans and used for training. The sets may be very large. Control systems may collect data from sensors as the systems operate and use that to identify parameters or train the system.

Models

Models are often used in learning systems. A model provides a mathematical framework for learning. A model is human derived and based on human observations and experiences. For example, a model of a car might be that it is rectangular shaped with dimensions that fit within a standard parking spot. Models

are usually thought of as human derived and providing a framework for machine learning. However, some forms of machine learning develop their own models without a human-derived structure.

Training

A system that maps an input to an output needs training to do this in a useful way. Just as people need to be trained to perform tasks, machine learning systems need to be trained. Training is accomplished by giving the system an input and the corresponding output and modifying the structure (models or data) in the learning machine so that mapping is learned. In some ways this is like curve fitting or regression. If we have enough training pairs, then the system should be able to produce correct outputs when new inputs are introduced. For example, if we give a face recognition system thousands of cat images and tell it that those are cats, we hope that when it is given new cat images, it will also recognize them as cats. Problems can arise when you don't give it enough training sets or the training data are not sufficiently diverse, that is, do not represent the full range of cats in this example.

Supervised Learning

Supervised learning means that specific training sets of data are applied to the system. The learning is supervised in that the "training sets" are human derived. It does not necessarily mean that humans are actively validating the results. The process of classifying the system's outputs for a given set of inputs is called labeling. That is, you explicitly say which results are correct or which outputs are expected for each set of inputs.

The process of generating training sets can be time consuming. Great care must be taken to ensure that the training sets will provide sufficient training so that when real-world data are collected the system will produce correct results. They must cover the full range of expected inputs and desired outputs. The training is followed by test sets to validate the results. If the results aren't good, then the test sets are cycled into the training sets and the process repeated.

A human example would be a ballet dancer trained exclusively in classical ballet technique. If she were then asked to dance a modern dance, the results might not be as good as required because the dancer did not have the appropriate training sets; her training sets were not sufficiently diverse.

Unsupervised Learning

Unsupervised learning does not utilize training sets. It is often used to discover patterns in data for which there is no "right" answer. For example, if you used unsupervised learning to train a face identification system, the system might cluster the data in sets, some of which might be faces. Clustering algorithms are generally examples of unsupervised learning. The advantage of unsupervised learning is that you can learn things about the data that you might not know in advance. It is a way of finding hidden structures in data.

Semisupervised Learning

With the semisupervised approach, some of the data is in the form of labeled training sets and other data are not. In fact, typically only a small amount of the input data is labeled while most is not, as the labeling may be an intensive process requiring a skilled human. The small set of labeled data is leveraged to interpret the unlabeled data.

Online Learning

The system is continually updated with new data. This is called "online" because many of the learning systems use data collected online. It could also be called "recursive learning." It can be beneficial to periodically "batch" process data used up to a given time and then return to the online learning mode.

Pytorch

PyTorch is a Python library that performs immediate execution of dynamic tensor computations with automatic differentiation and GPU acceleration and does so while maintaining performance comparable to the fastest current libraries for deep learning. PyTorch builds on these trends by providing an array-based programming model accelerated by GPUs and differentiable via automatic differentiation integrated in the Python ecosystem.

PyTorch maintains a strict separation between its control (i.e. program branches, loops) and data flow (i.e. tensors and the operations performed on them). The resolution of the control flow is handled by Python and optimized C++ code executed on the host CPU, and result in a linear sequence of operator invocations on the device. Operators can be run either on CPU or on GPU.

PyTorch is designed to execute operators asynchronously on GPU by leveraging the CUDA stream mechanism to queue CUDA kernel invocations to the GPUs hardware FIFO. This allows the system to overlap the execution of Python code on CPU with tensor operators on GPU. Because the tensor operations usually take a significant amount of time, this lets us saturate the GPU and reach peak

performance even in an interpreted language with fairly high overhead like Python. This mechanism is nearly invisible to the user. Unless they implement their own multi-stream primitives all of the CPU-GPU synchronization is handled by the library.

PyTorch could leverage a similar mechanism to also execute operators asynchronously on the CPU. However, the costs of cross-thread communication and synchronization would negate the performance benefit of such an optimization.

5.4.2.3 Datasets

The development of robust autonomous driving models depends on having access to large-scale training datasets, especially as more learning-based approaches are incorporated. Over the past decade, tens of datasets for autonomous driving have been collected and made public by multiple institutes around the world. These datasets are a valuable resource for the research community to develop benchmarks and consolidate research efforts.

We have chosen nuScenes and Argoverse datasets to train our model. The largest dataset that provides the most sensor measurements is nuScenes, which contains 1000 20-second-long videos with LiDAR, Radar, camera, IMU and GPS data. It also provides 3D bounding boxes over 25 classes of objects annotated at 2Hz. In the other hand, Argoverse includes sensor data collected by a fleet of autonomous vehicles in Pittsburgh and Miami as well as 3D tracking annotations, 300k extracted interesting vehicle trajectories, and rich semantic maps. The sensor data consists of 360◦ images from 7 cameras with overlapping fields of view, forward-facing stereo imagery, 3D point clouds from long range LiDAR, and 6-dof pose.

nuScenes dataset

nuScenes represents a large leap forward in terms of data volumes and complexities and is the first dataset to provide 360◦ sensor coverage from the entire sensor suite. It is also the first AV dataset to include radar data and captured using an AV approved for public roads. It is further the first multimodal dataset that contains data from nighttime and rainy conditions, and with object attributes and scene descriptions in addition to object class and location. nuScenes is a holistic scene understanding benchmark for AVs. It enables research on multiple tasks such as object detection, tracking and behavior modeling in a range of conditions .

There has been publishing the devkit, evaluation code, taxonomy, annotator instructions, and database schema for industry wide standardization. Recently, the Lyft L5 [98] dataset adopted this format to achieve compatibility between the different datasets. The nuScenes data is published under CC BY-NC-SA 4.0 license, which means that anyone can use this dataset for non-commercial research purposes. All data, code, and information is made available online.

Here the description on how nuScenes planned drives, set upped the vehicles, selected interesting scenes, annotated the dataset and protected the privacy of third parties.

Drive planning

The vehicles drive in Boston (Seaport and South Boston) and Singapore (One North, Holland Village and Queenstown), two cities that are known for their dense traffic and highly challenging driving situations. The emphasizing was on the diversity across locations in terms of vegetation, buildings, vehicles, road markings and right versus left-hand traffic. From a large body of training data they manually select 84 logs with 15h of driving data (242km travelled at an average of 16km/h). Driving routes are carefully chosen to capture a diverse set of locations (urban, residential, nature and industrial), times (day and night) and weather conditions (sun, rain and clouds).

Sensor Details

The list of the sensors used to create the nuScenes dataset is as bellow:

- 6x Camera: RGB, 12Hz capture frequency, 1/1.8" CMOS sensor, 1600 × 900 resolution, auto exposure, JPEG compressed
- 1x Lidar: Spinning, 32 beams, 20Hz capture frequency, 360◦ horizontal FOV, −30◦ to 10◦ vertical FOV, ≤ 70m range, ±2cm accuracy, up to 1.4M points per second.
- 5x Radar ≤ 250m range, 77GHz, FMCW, 13Hz capture frequency, ±0.1km/h vel. accuracy
- GPS & IMU: GPS, IMU, AHRS. 0.2◦ heading, 0.1◦ roll/pitch, 20mm RTK positioning, 1000Hz update rate

Car setup

They used two Renault Zoe supermini electric cars with an identical sensor layout to drive in Boston and Singapore. Front and side cameras have a 70◦ FOV and are offset by 55◦. The rear camera has a FOV of 110◦. Sensor synchronization. To achieve good cross-modality data alignment between the lidar and the cameras, the exposure of a camera is triggered when the top lidar sweeps across the center of the camera's FOV. The timestamp of the image is the exposure trigger time; and the timestamp of the lidar scan is the time when the full rotation of the current lidar frame is achieved. Given that the camera's exposure time is nearly instantaneous, this method generally yields good data alignment5. They perform motion compensation using the localization algorithm described below.

Localization

Most existing datasets provide the vehicle location based on GPS and IMU. Such localization systems are vulnerable to GPS outages. As they operate in dense urban areas, this problem is even more pronounced. To accurately localize the vehicle, they create a detailed HD map of lidar points in an offline step. While collecting data, they use a Monte Carlo Localization scheme from lidar and odometry information. This method is very robust, and they achieve localization errors of ≤ 10cm. To encourage robotics research, they also provide the raw CAN bus data (e.g. velocities, accelerations, torque, steering angles, wheel speeds).

Maps

They provide highly accurate human-annotated semantic maps of the relevant areas. The original rasterized map includes only roads and sidewalks with a resolution of 10px/m. The vectorized map expansion provides information on 11 semantic classes, making it richer than the semantic maps of other datasets published since the original release. The cameras run at 12Hz while the lidar runs at 20Hz. The 12 camera exposures are spread as evenly as possible across the 20 lidar scans, so not all lidar scans have a corresponding camera frame. Finally, they provide the baseline routes - the idealized path an AV should take, assuming there are no obstacles. This route may assist trajectory prediction, as it simplifies the problem by reducing the search space of viable routes.

Scene selection

After collecting the raw sensor data, they manually select 1000 interesting scenes of 20s duration each. Such scenes include high traffic density (e.g. intersections, construction sites), rare classes (e.g. ambulances, animals), potentially dangerous traffic situations (e.g. jay-walkers, incorrect behavior), maneuvers (e.g. lane change, turning, stopping) and situations that may be difficult for an AV. They also select some scenes to encourage diversity in terms of spatial coverage, different scene types, as well as different weather and lighting conditions. Expert annotators write textual descriptions or captions for each scene (e.g.: “Wait at intersection, peds on sidewalk, bicycle crossing, jaywalker, turn right, parked cars, rain”).

Data annotation

Having selected the scenes, they sample keyframes (image, lidar, radar) at 2Hz. They annotate each of the 23 object classes in every keyframe with a semantic category, attributes (visibility, activity, and pose) and a cuboid modeled as x, y, z, width, length, height and yaw angle. They annotate objects continuously throughout

each scene if they are covered by at least one lidar or radar point. Using expert annotators and multiple validation steps, they achieve highly accurate annotations. They also release intermediate sensor frames, which are important for tracking, prediction and object detection. At capture frequencies of 12Hz, 13Hz and 20Hz for camera, radar and nlidar, this makes this dataset unique. Only the Waymo Open dataset provides a similarly high capture frequency of 10Hz.

Annotation statistics

nuScenes dataset has 23 categories including different vehicles, types of pedestrians, mobility devices and other objects. They present statistics on geometry and frequencies of different classes. Per keyframe there are 7 pedestrians and 20 vehicles on average. Moreover, 40k keyframes were taken from four different scene locations (Boston: 55%, SG-OneNorth: 21.5%, SG-Queenstown: 13.5%, SG-HollandVillage: 10%) with various weather and lighting conditions (rain: 19.4%, night: 11.6%). Due to the finegrained classes in nuScenes, the dataset shows severe class imbalance with a ratio of 1:10k for the least and most common class annotations (1:36 in KITTI). This encourages the community to explore this long tail problem in more depth.

Annotated objects contain up to 100 lidar points even at a radial distance of 80m and at most 12k lidar points at 3m. At the same time they contain up to 40 radar returns at 10m and 10 at 50m. The radar range far exceeds the lidar range at up to 200m.

Argoverse dataset

Argoverse includes sensor data collected by a fleet of autonomous vehicles in Pittsburgh nand Miami as well as 3D tracking annotations, 300k extracted interesting vehicle trajectories, and rich semantic maps. The sensor data consists of 360◦ images from 7 cameras with overlapping fields of view, forward-facing stereo imagery, 3D point clouds from long range LiDAR, and 6-dof pose. 290km of mapped lanes contain rich geometric and semantic metadata which are not currently available in any public dataset.

Argoverse sensor data, maps, and annotations are the primary contribution of this work. They also develop an API which helps connect the map data with sensor information e.g. ground point removal, nearest centerline queries, and lane graph connectivity; see Supplemental Material for more details. They collect raw data from a fleet of autonomous vehicles in Pittsburgh, Pennsylvania, USA and Miami, Florida, USA. These cities have distinct climate, architecture, infrastructure, and behavior patterns. The captured data spans different seasons, weather conditions,

and times of day. The data used for this dataset traverses nearly 300km of mapped road lanes and comes from a subset of the fleet operating area.

Sensors

The cars are equipped with two roof-mounted VLP-32 LiDAR sensors with an overlapping 40◦ vertical field of view and a range of 200m, roughly twice that as the sensors used in nuScenes and KITTI. On average, the LiDAR sensors produce a point cloud at each sweep with three times the density of the LiDAR sweeps in the nuScenes dataset (Argoverse $\sim$ 107, 000 points vs. nuScenes' $\sim$ 35, 000 points). The vehicles have 7 high-resolution ring cameras (1920 × 1200) recording at 30 Hz with overlapped field of view providing 360◦ coverage. In addition, there are 2 front-facing stereo cameras (2056×2464) sampled at 5 Hz. Faces and license plates are procedurally blurred in camera data to maintain privacy. Finally, 6-DOF localization for each timestamp comes from a combination of GPS-based and sensor-based localization. Vehicle localization and maps use a city-specific coordinate system described in more detail in the Supplemental Material. Sensor measurements for particular driving sessions are stored in "logs", and they provide intrinsic and extrinsic calibration data for the LiDAR sensors and all 9 cameras for each log. They place the origin of the vehicle coordinate system at the center of the rear axle. All sensors are roof-mounted, with a LiDAR sensor surrounded by 7 "ring" cameras (clockwise: facing front center, front right, side right, rear right, rear left, side left, and front left) and 2 stereo cameras.

Maps

Argoverse contains three distinct maps – (1) a vector map of lane centerlines and their attributes, (2) a rasterized map of ground height, and (3) a rasterized map of driveable area and region of interest (ROI).

Vector Map of Lane Geometry

The vector map consists of semantic road data represented as a localized graph rather than rasterized into discrete samples. The vector map is a simplification of the map used in fleet operations. In the vector map, there are lane centerlines, split into lane segments. They observe that vehicle trajectories generally follow the center of a lane so this is a useful prior for tracking and forecasting.

A lane segment is a segment of road where cars drive in single-file fashion in a single direction. Multiple lane segments may occupy the same physical space (e.g. in an intersection). Turning lanes which allow traffic to flow in either direction would be represented by two different lanes that occupy the same physical space.

For each lane centerline, they provide a number of semantic attributes. These lane attributes describe whether a lane is located within an intersection or has an associated traffic control measure (Boolean values that are not mutually inclusive). Other semantic attributes include the lane's turn direction (left, right, or none) and the unique identifiers for the lane's predecessors (lane segments that come before) and successors (lane segments that come after) of which there can be multiple (for merges and splits, respectively). Centerlines are provided as "polylines", i.e. a sequence of straight segments. Each straight segment is defined by 2 vertices: (x, y, z) start and (x, y, z) end. Thus, curved lanes are approximated with a set of straight lines.

In Miami, lane segments that could be used for route planning are on average 3.84m ± 0.89 wide. In Pittsburgh, the average width is 3.97m ± 1.04 in width. Other types of lane segments that would not be suitable for self-driving, e.g. bike lanes, can be as narrow as 0.97m in Miami and as narrow as 1.06m in Pittsburgh.

Rasterized Drivable Area Map

The maps include binary drivable area labels at 1 meter grid resolution. A drivable area is an area where it is possible for a vehicle to drive (though not necessarily legal). Drivable areas can encompass a road's shoulder in addition to the normal drivable area that is represented by a lane segment. The track annotations extend to 5 meters beyond the drivable area. They call this larger area the region of interest (ROI).

Rasterized Ground Height Map

Finally, the maps include real-valued ground height at 1 meter resolution. Knowledge of ground height can be used to remove LiDAR returns on static ground surfaces and thus makes the 3D detection of dynamic objects easier.

3D Track Annotations

Argoverse-Tracking-Beta1 contains 100 vehicle log segments with human-annotated data 3D tracks. These 100 segments vary in length from 15 to 60 seconds and collectively contain 10,572 tracked objects. For each log segment, they annotate all objects of interest (both dynamic and static) with bounding cuboids which follow the 3D LiDAR returns associated with each object over time. They only annotate objects within 5 meters of the drivable area as defined by the map. For objects that are not visible for the entire segment duration, tracks are instantiated as soon as the object becomes visible in the LiDAR point cloud and

tracks are terminated when the object ceases to be visible. They mark objects as "occluded" whenever they become invisible within the sequence. Each object is labeled with one of 17 categories, including OTHER_STATIC and OTHER_MOVER for static and dynamic objects that do not fit into other predefined categories. More than 70% of tracked objects are vehicles, but they also observe pedestrians, bicycles, mopeds, and more. All track labels pass through a manual quality assurance review process. They divide the annotated tracking data into 60 training, 20 validation, and 20 testing sequences.

Mined Trajectories for Motion Forecasting

They are also interested in studying the task of motion forecasting in which they predict the location of a tracked object sometime in the future. Motion forecasts can be critical to safe autonomous vehicle motion planning. While the human-annotated 3D tracks are suitable training and test data for motion forecasting, the motion of many of vehicles is relatively uninteresting – in a given frame, most cars are either parked or traveling at nearly constant velocity. Such tracks are hardly a representation of real forecasting challenges. They would like a benchmark with more diverse scenarios e.g. managing an intersection, slowing for a merging vehicle, accelerating after a turn, stopping for a pedestrian on the road, etc. To sample enough of these interesting scenarios, they track objects from 1006 driving hours across both Miami and Pittsburgh and find vehicles with interesting behavior in 320 of those hours. In particular, they look for vehicles that are either (1) at intersections (2) taking left or right turns (3) changing to adjacent lanes or (4) in dense traffic. In total, they collect 333,441 five second sequences and use them in the forecasting benchmark. Each sequence contains the 2D, birds-eye-view centroid of each tracked object sampled at 10hz. The 333,441 sequences are split into 211,691 train, 41,146 validation, and 80,604 test sequences. Each sequence has one challenging trajectory which is the focus of the forecasting benchmark. The train, val, and test sequences are taken from disjoint parts of the cities, i.e., roughly one eighth and one quarter of each city is set aside as validation and test data, respectively. This dataset is far larger than what could be mined from publicly available autonomous driving datasets, and they have the advantage of using the maps to make it easier to track objects. While data of this scale is appealing because it allows to see rare behaviors and train complex models, it is too large to exhaustively verify the accuracy of the mined trajectories and thus there is some noise and error inherent in the data.

5.4.3 The embedded platform

We targeted NVIDIA Jetson AGX Xavier that is representative of the next-generation AV Domain Controller as our embedded platform. This embedded platform has a GPGPU of 512-core Volta along with Tensor Core and a CPU of ARM 8-core v8.2 64-bit and would be a suitable choice for our system (for further description please refer to chapter 2).

5.4.4 The results

To evaluate our model, we use the standard metrics on the nuScenes leaderboard. The minimum average displacement error (ADE) over the top K predictions (MinADEK), the miss rate (MissRateK,2) only penalizes predictions that are further than 2 m from the ground truth, the offroad rate measures the fraction of predictions that are off the road, and for a set of k predictions for each agent a in a scene, we report the minimum Final Displacement Error (minFDEk) to the ground truth. Since all examples in nuScenes are on the road, this should be zero. The minimum Average Displacement Error computes the L2 norm between's and the closest joint prediction. The minimum Final Displacement Error is equivalent to evaluating the minADE at a single time step T. The overlap rate is computed by taking the highest confidence joint prediction from each multimodal joint prediction. If any of the A agents in the jointly predicted trajectories overlap at any time with any other objects that were visible at the prediction time step (compared at each time step up to T) or with any of the jointly predicted trajectories, it is considered a single overlap. The overlap rate is computed as the total number of overlaps divided by the total number of predictions. The overlap is calculated using box intersection, with headings inferred from consecutive waypoint position differences. The formulation of minADE and minFDE can derived from [106]:

$$minADE_k = min_{m=\{1,2,\ldots,k\}} \frac{1}{t_f} \sum_t^{t_f} \|\hat{y}_t^m + y_t\|_2 \quad (1)$$

$$minFDE_k = min_{m=\{1,2,\ldots,k\}} \frac{1}{t_f} \sum_t^{t_f} \left\| \hat{y}_{t_f}^m + y_{t_f} \right\|_2 \quad (2)$$

where *k* denotes the number of modalities that are the most probable trajectories according to the estimated scores. *minADE* and *minFDE* are the average of all target agents in the dataset.

Although these metrics are widely used in the motion forecasting task, minADE and minFDE only depend on the error from the ground truth; therefore, it is

impractical to evaluate whether the predictions are realistic. Several works have thus proposed new metrics in addition to ADE/FDE to measure the feasibility of the outputs. The Off-Road Rate [107] evaluates the feasibility of multiple outputs by calculating the percentage of the outputs that lie out of drivable area. Finally MissRate (MR) can be calculated as follows:

$$MissRate(\hat{S}) = \frac{|\hat{S}|+|\hat{S}_{hit}|}{|\hat{S}|} \quad (3)$$

where $|\hat{S}|$ is the number of predicted and matched agents which have complete future trajectories across different scenes. $|\hat{S}_{hit}|$ is the number of hitted agents which belong to $\hat{S}$.

Since our work is designed to outperform the predictions specifically in the intersections, we require to consider the works that are considering the intersections and compare our work with them. We demonstrate our results on some nuScenese leaderboard showing on table 2, comparing with the top performing entries on the nuScenes leaderboard. We achieve state of the art results on some metrics in this specific area.

Table 2. The comparison of the motion prediction models from the nuScenes leaderboard that consider the intersections.

Model	**MinADE (5)**	**MinADE (10)**	**MissRate Top-5 (2m)**	**MissRate Top-10 (2m)**	**MinFDE (1)**	**Off Road Rate**
Trajectron++ [91]	1.88	1.51	0.70	0.57	9.52	0.25
Autobot [99]	1.43	1.05	0.66	0.45	8.66	**0.03**
P2T [100]	1.45	1.16	0.64	0.46	10.50	**0.03**
PGP [101]	**1.30**	**1.00**	0.61	**0.37**	7.17	**0.03**
Our model	1.37	**1.00**	**0.53**	0.41	**6.59**	0.05

As it can be seen in Table 2, we have outperformed in MinFDE and MissRate. This is because the traffic rules help the Final Displacement Error to be less than the times of not considering the traffic rules. Traffic rules help to predict the final positions as better as it is possible. Therefore, we have outperformed in MinFDE. We have also competitive results with the state-of-the-art works in MinADE. Our contribution was to outperform the motion prediction model at the intersections as shown in Figure 35.

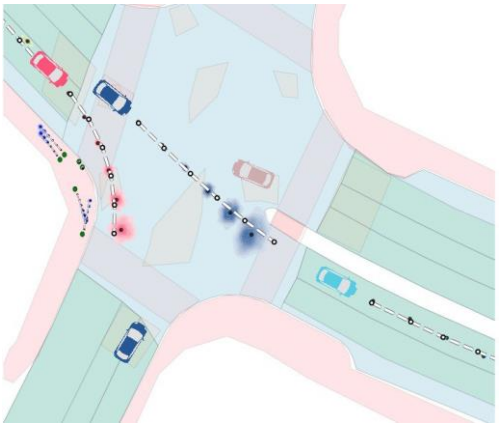

Fig. 35. One example of our implementation in the nuScenes dataset at the intersection.

5.5 Conclusion

In this chapter, the new method of the motion prediction has been discussed. Furthermore, two most recent works have been implemented with the results on Carla simulator. We have discussed the motion prediction in the intersections and demonstrated that if we include the traffic rules into the model, we can achieve and outperformance respect to the other works.

Including the traffic rules into the models for the motion prediction can enhance the metrics of the motion prediction and outperforms the previous works. In this regard, we have included the traffic rules in the intersections and by comparing it with the similar works, we have identified the better performance.

The traffic rules are not limited to the intersections, and it can be extended to the other traffic rules in the round abouts, entering and exiting lines, and the other ones. By activating each traffic rule and including it in the model, we can outperform the previous works in that area. Therefore, the next works are considering the other traffic rules in the other areas and including them in the model.

After considering the other traffic rules, we can have a unified model that includes the implemented traffic rules in the model, and we can use that model in the different conditions. Indeed, combining this methodology with an HMI for the next movements would be another possible work to complete the usage of motion forecasting method for the motion sickness mitigation.

Overall, the new area that has been opened by our novel research of having the traffic rules in the models not only improves the results of the motion prediction model, but also opens a new area of researching with different traffic rules in various conditions and places. It is very important to remind that if the traffic rules increase in a model, we would require a more powerful embedded platform to execute the model in real-time. Therefore, a new research area will be also using the new embedded platforms that are appropriate for this work.

Chapter 6

Conclusion and Future Works

In this thesis, I introduced my research work as the title of Advanced Driver Assistance Systems for high-performance Embedded Automotive Platforms. I started with the first chapter of Introduction to Advanced Driver Assistance Systems. In this chapter, I went through different ADASs and introduced the main requirements for having an ADAS. At the end of this chapter, I introduced the ADASs that I developed. Since one of the main required components in designing the ADASs is their embedded platform, in the second chapter, I introduced the high-performance automotive embedded platforms, and I described what embedded platform I used for the implementation of my work. Then, in the third chapter, I described A Full-Featured, Enhanced Cost Function to Mitigate Motion Sickness in Semi- and Fully-autonomous Vehicles. In this chapter, we focused on the five main physical characteristics that affect motion sickness, including them in the function cost, to provide quality passengers' experience to vehicle passengers. We implemented our approach in a state-of-the-art Model Predictive Controller, to be used in a real Autonomous Vehicle. The potential sources of AV motion sickness can be divided into five groups, namely, variation in horizontal and vertical acceleration, posture instability, loss of controllability and loss of anticipation of motion direction, Head downward inclination, and lack of synchronization between virtual motion and the vehicle motion profile. In this work, which I presented in VEHITS 2021, we focused on the three sources, namely, variation in horizontal and vertical acceleration, posture instability, and loss of controllability. Anyway, in the fourth chapter, I described the second research work for motion sickness mitigation considering the other sources, lack of synchronization between virtual motion and the vehicle motion profile and loss of anticipation of motion direction. In this chapter I described Motion Sickness Minimization Alerting System Using The Next Curvature Topology that tries to interact with the passengers for preventing the motion sickness. I presented this research work in IEEE ICMA 2022. Finally, to conclude the thesis, I introduced the last research work on the motion prediction focusing on the traffic rules injection in the intersection. I started the chapter of Motion Prediction using Attention Heads and Traffic rules in intersections by two

tryout of implementation of the state-of-the-art works and then introduced my new approach for outperforming the previous works in the intersections. Therefore, we demonstrated that by injecting the traffic rules in the model we can have even better results. This work is submitted in IEEE ICMA 2023.

The research work that I did, started from researching about the comfort issues of the autonomous driving and led me to introduce two different work to mitigate the motion sickness. Researching on the motion sickness also pursued me to start researching on the motion prediction and proposing a novel method for outperforming in the intersections.

In the other hand, these research works have opened a window to the future works. The most interesting future works can be started by implementation the first two methods of the motion sickness in the real vehicle and facing the real vehicle issues. To do so, we need to consider also the more complex vehicle models, such as the kinematic and dynamic vehicle model, to validate our approach at highest speeds (i.e., > 150km/h), and to possibly include other classes of vehicles, such as busses and coaches, which potentially issue Motion Sickness much more than cars using the online services like google maps to give us the best results real-time.

Furthermore, the mixture of the motion prediction model with the motion sickness mitigation systems can be a very interesting and challenging area to follow. Since with a better prediction model, we will have a better interactive system with the passengers in real-time. To do so, we need to extend our model to the other traffic rules. It also requires a research work on a higher performance embedded platform to be able to execute the model in real-time.

References

[1] World Health Organization. (2021). Global status report on road safety. WHO. Geneva, Switzerland.

[2] Association for Safe International Road Travel. (2018). Annual global road crash statistics. ASIRT. Potomac, Maryland. [Online]. Available: http://asirt.org/initiatives/in-forming-road-users/road-safety-cts/roadcrash-statistics

[3] B. Markwalter, "The path to driverless cars," IEEE Consum. Electron. Mag., vol. 6, no. 2, pp. 125–126, 2017.

[4] W. Cunningham. (2014, May 31). US requiring back-up cameras in cars by 2018. Roadshow. [Online]. Available: www.cnet.com/news/u-srequiring-back-up-cameras-in-cars-by-2018/

[5] I. Gat, M. Benady, and A. Shashua, "A monocular vision advance warning system for the automotive aftermarket," SAE, Warrendale, PA, Tech. Rep. 2005-01-1470, 2005.

[6] P. Morignot, J. P. Rastelli, and F. Nashashibi, "Arbitration for balancing control between the driver and ADAS systems in an automated vehicle: Survey and approach," in Proc. IEEE Intelligent Vehicles Symp., 2014, pp. 575–580.

[7] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, and K. Lau, "Stanley: The robot that won the DARPA grand challenge," J. Field Robotics, vol. 23, no. 9, pp. 661–692, Sept. 2006.

[8] E. Guizzo, "How Google's self-driving car works," IEEE Spectr., vol. 18, no. 7, pp. 1132–1141, 2011.

[9] Velodyne. (2017). Velodyne LiDAR announces new velarray LiDAR sensor. Business Wire. [Online]. Available: http://www.businesswire .com/news/home/20170419005516/en/

[10] NXP. (2017, May 24). RADAR, camera, LiDAR and V2X for autonomous cars. [Online]. Available: https://blog.nxp.com/automotive/ radar-camera-and-lidar-for-autonomous-cars

[11] S. Saponara. (2016). Hardware accelerator IP cores for real time radar and camera-based ADAS. J. Real-Time Image Processing. [Online]. pp. 1–18. Available: https://doi.org/10.1007/s11554-016-0657-0

[12] D. I. B Hagebeuker. (2007). A 3D time of flight camera for object detection. PMD Technologies GmbH. Siegen, Germany. [Online]. Available: https://www.ifm.com/obj/O1D_Paper-PMD.pdf

[13] P. Viola and M. J. Jones, "Robust real-time face detection," Int. J. Comput. Vision, vol. 57, no. 2, pp. 137–154, 2004.

[14] J. Pyo, J. Bang, and Y. Jeong, "Front collision warning based on vehicle detection using CNN," in Proc. IEEE SoC Design Conf. (ISOCC), 2016, pp. 163–64.

[15] Y. Huang, Y. Zhang, Z. Wu, N. Li, and J. Chambers, "A novel adaptive Kalman filter with inaccurate process and measurement noise covariance matrices," IEEE Trans. Autom. Control, vol. 63, no. 2, pp. 594–601, 2018.

[16] S. Nedevschi, R. Danescu, D. Frentiu, T. Marita, F. Oniga, C. Pocol, R. Schmidt, and T. Graf, "High accuracy stereo vision system for far distance obstacle detection," in Proc. IEEE Intelligent Vehicles Symp., pp. 292–297, 2004.

[17] S. Diamantas, S. Astaras, and A. Pnevmatikakis, "Depth estimation in7 still images and videos using a motionless monocular camera," in Proc. IEEE Int. Conf. Imaging Systems Techniques (IST), 2016, pp. 129–134.

[18] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 7, pp. 1239–1258, 2010.

[19] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in Proc. IEEE Conf. Computer Vision Pattern Recognition (CVPR), 2017, pp. 6517–6525.

[20] F. Parada-Loira and J. L. Alba-Castro, “Local contour patterns for fast traffic sign detection,” in Proc. IEEE Intelligent Vehicles Symp. (IV), 2010, pp. 1–6.

[21] Y. Li, L. Chen, H. Huang, X. Li, W. Xu, L. Zheng, and J. Huang, “Nighttime lane markings recognition based on canny detection and Hough transform,” in Proc. IEEE Int. Conf. Real-Time Computing Robotics (RCAR), 2016, pp. 411–415.

[22] S. G. Klauer, T. A. Dingus, V. L. Neale, J. D. Sudweeks, and J. D. Ramsey. (2004, Jan.). The impact of driver inattention on near-crash/ crash risk: An analysis using the 100-car naturalistic driving study data. U.S. Nat. Highway Traffic Safety Admin. Washington, D.C. [Online]. Available: https://www.nhtsa.gov/DOT/NHTSA/NRD/Articles/HF/Reducing%20Unsafe%20behaviors/810594/810594.htm

[23] M. I. Chacon-Murguia and C. Prieto-Resendiz, “Detecting driver drowsiness: A survey of system designs and technology,” IEEE Consum. Electron. Mag., vol. 4, no. 4, pp. 107–119, 2015.

[24] F. Castanedo. (2013). A review of data fusion techniques. Sci. World J. [Online]. Available: http://dx.doi.org/10.1155/2013/704504

[25] K. Abboud, H. A. Omar, and W. Zhuang, “Interworking of DSRC and cellular network technologies for V2X communications: A survey,” IEEE Trans. Veh. Technol., vol. 65, no. 12, 2016, pp. 9457–9470.

[26] N. Vivek, S. V. Srikanth, P. Saurabh, T. P. Vamsi, and K. Raju, “On field performance analysis of IEEE 802.11 P and WAVE protocol stack for V2V & V2I communication,” in Proc. IEEE Int. Conf. Information Communication Embedded Systems (ICICES), 2014, pp. 1–6.

[27] SAE International. [Online]. Available: http://www.sae.org/misc/pdfs/automated_driving.pdf

[28] C. Schwarzlmuller, F. Al Machot, A. Fasih, and K. Kyamakya, “Adaptive contrast enhancement involving CNN-based processing for foggy weather conditions & non-uniform lighting conditions,” in Proc. IEEE Int.Symp. Theoretical Electrical Engineering (ISTET), 2011, pp. 1–10.

[29] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, “Experimental security analysis of a modern automobile,” in Proc. IEEE Security Privacy, 2010, pp. 447–462.

[30] C. Valasek and C. Miller. (2015). Remote exploitation of an unaltered passenger vehicle. IOActive. Seattle, WA. White Paper. [Online]. Available: https://ioactive.com/pdfs/IOActive_Remote_Car_Hacking.pdf

[31] Iskander, J., Attia, M., Saleh, K., Nahavandi, D., Abobakr, A., Mohamed, S., ... & Hossny, M. (2019). From car sickness to autonomous car sickness: A review. Transportation research part F: traffic psychology and behaviour, 62, 716-726.

[32] S. Saidi, S. Steinhorst, A. Hamann, D. Ziegenbein, and M. Wolf. Special session: Future automotive systems design: Research challenges and opportunities. In International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pages 1–7, 2018.

[33] R. Cavicchioli, N. Capodieci, and M. Bertogna. Memory interference characterization between CPU cores and integrated GPUs in mixedcriticality platforms. In 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pages 1–10, 2017.

[34] Mohamed Hassan and Rodolfo Pellizzoni. ”Bounding DRAM interference in COTS heterogeneous MPSoCs for mixed criticality systems”. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 37(11):2323–2336, 2018.

[35] F. Restuccia, M. Pagani, A. Biondi, M. Marinoni, and G. Buttazzo. ”Modeling and analysis of bus contention for hardware accelerators in FPGA SoCs.” 32nd Euromicro Conference on Real-Time Systems (ECRTS 2020). Schloss Dagstuhl-Leibniz-Zentrum f¨ur Informatik, 2020.

[36] D. Casini, A. Biondi, G. Nelissen, and G. Buttazzo. ”A holistic memory contention analysis for parallel real-time tasks under partitioned scheduling”. In Proceedings of the 26th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2020), 2020.

[37] Arm® Architecture Reference Manual Supplement Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A, available online at https://developer.arm.com/docs/ddi0598/latest

[38] A. Hamann, S. Saidi, D. Ginth¨or, C. Wietfeld, and D. Ziegenbein. Building End-to-End IoT Applications with QoS Guarantees. In 2020 57th ACM/IEEE Design Automation Conference (DAC), 2020.

[39] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha. MemGuard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms. In Real-Time and Embedded Technology and Applications Symposium (RTAS), pages 55–64, 2013.

[40] I. Moazen and P. Burgio, "A Full-Featured, Enhanced Cost Function to Mitigate Motion Sickness in Semi-and Fully-autonomous Vehicles", 2021, bll 497–504.

[41] Moazen, I., Burgio, P., & Castellano, A. (2022, August). Motion Sickness Minimization Alerting System Using The Next Curvature Topology. In 2022 IEEE International Conference on Mechatronics and Automation (ICMA) (pp. 635-640). IEEE.

[42] J. Elsner, "Optimizing passenger comfort in cost functions for trajectory planning," arXiv.org, 16-Nov-2018. [Online]. Available: https://arxiv.org/abs/1811.06895.

[43] H. Bellem, B. Thiel, M. Schrauf, en J. F. Krems, "Comfort in automated driving: An analysis of preferences for different automated driving styles and their dependence on personality traits", Transp. Res. Part F Traffic Psychol. Behav., vol 55, bll 90–100, Mei 2018.

[44] Diels, C. (2014). Will autonomous vehicles make us sick? Contemporary ergonomics and human factors. Taylor & Francis, 301-307.

[45] Sivak, M., & Schoettle, B. (2015). Motion sickness in self-driving vehicles. University of Michigan, Ann Arbor, Transportation Research Institute.

[46] Donohew, B. E., & Griffin, M. J. (2004). Motion sickness: effect of the frequency of lateral oscillation. Aviation, Space, and Environmental Medicine, 75(8), 649-656.

[47] B. S. I. Standard, 6841, Guide to Measurement and Evaluation of Human Exposure to Whole-Body Mechanical Vibration and Repeated Shock. London, UK: BSI, 1987.

[48] MATLAB. (2020). version (R2020a). Natick, Massachusetts: The MathWorks Inc.

[49] Mechanical Simulation Corporation. (2020). Introduction to CarSim.

[50] Epic Games. (2019). Unreal Engine. Retrieved from https://www.unrealengine.com

[51] Reason, J. T., & Graybiel, A. (1969). Changes in subjective estimates of well-being during the onset and remission of motion sickness symptomatology in the slow rotation room (Vol. 1083). US Naval Aerospace Medical Institute, Naval Aerospace Medical Center.

[52] Lambert, E., Romano, R., & Watling, D. (2019, May). Optimal path planning with clothoid curves for passenger comfort. In Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2019) (Vol. 1, pp. 609-615). SciTePress.

[53] Htike, Z., Papaioannou, G., Velenis, E., & Longo, S. (2020, May). Motion Planning of Self-driving Vehicles for Motion Sickness Minimisation. In 2020 European Control Conference (ECC) (pp. 1719-1724). IEEE.

[54] N. M. Yusof, J. Karjanto, J. M. B. Terken, F. L. M. Delbressine, en G. W. M. Rauterberg, “Gaining situation awareness through a vibrotactile display to mitigate motion sickness in fully-automated driving cars”, International Journal of Automotive and Mechanical Engineering, vol 17, no 1, bll 7771–7783, 2020.

[55] R. Hainich, U. Drewitz, K. Ihme, J. Lauermann, M. Niedling, en M. Oehl, “Evaluation of a human–machine interface for motion sickness mitigation utilizing anticipatory ambient light cues in a realistic automated driving setting”, Information (Basel), vol 12, no 4, bl 176, Apr 2021.

[56] O. X. Kuiper, J. E. Bos, C. Diels, en E. A. Schmidt, “Knowing what’s coming: Anticipatory audio cues can mitigate motion sickness”, Appl. Ergon., vol 85, no 103068, bl 103068, Mei 2020.

[57] K. N. de Winkel, P. Pretto, S. A. E. Nooij, I. Cohen, en H. H. Bülthoff, “Efficacy of augmented visual environments for reducing sickness in autonomous vehicles”, Appl. Ergon., vol 90, no 103282, bl 103282, Jan 2021.

[58] Easa, S. M., & Diachuk, M. (2020). Optimal Speed Plan for the Overtaking of Autonomous Vehicles on Two-Lane Highways. Infrastructures, 5(5), 44.

[59] Shi, J., Sun, D., Qin, D., Hu, M., Kan, Y., Ma, K., & Chen, R. (2020). Planning the trajectory of an autonomous wheel loader and tracking its trajectory via adaptive model predictive control. Robotics and Autonomous Systems, 103570.

[60] Wu, J., Wang, Z., & Zhang, L. (2020). Unbiased-estimation-based and computation-efficient adaptive MPC for four-wheel-independently-actuated electric vehicles. Mechanism and Machine Theory, 154, 104100.

[61] Luan, Z., Zhang, J., Zhao, W., & Wang, C. (2020). Trajectory Tracking Control of Autonomous Vehicle with Random Network Delay. IEEE Transactions on Vehicular Technology.

[62] Geng, K., & Liu, S. (2020). Robust Path Tracking Control for Autonomous Vehicle Based on a Novel Fault Tolerant Adaptive Model Predictive Control Algorithm. Applied Sciences, 10(18), 6249.

[63] Kang, C. M., Lee, S. H., & Chung, C. C. (2014, October). Lane estimation using a vehicle kinematic lateral motion model under clothoidal road constraints. In 17th International IEEE Conference on Intelligent Transportation Systems (ITSC) (pp. 1066-1071). IEEE.

[64] Rajamani, R. (2011). Vehicle dynamics and control. Springer Science & Business Media.

[65] Antonelli, D., Nesi, L., Pepe, G., & Carcaterra, A. (2019, June). A novel approach in Optimal trajectory identification for Autonomous driving in racetrack. In 2019 18th European Control Conference (ECC) (pp. 3267-3272). IEEE.

[66] Filip, J. (2018). Trajectory Tracking for Autonomous Vehicles.

[67] Muske, K. R., & Rawlings, J. B. (1993). Model predictive control with linear models. AIChE Journal, 39(2), 262-287.

[68] Önkol, M., & Kasnakoğlu, C. (2018). Adaptive model predictive control of a two-wheeled robot manipulator with varying mass. Measurement and Control, 51(1-2), 38-56.

[69] Hajizadeh, I., Hobbs, N., Sevil, M., Rashid, M., Askari, M. R., Brandt, R., & Cinar, A. (2020, May). Performance Monitoring, Assessment and Modification of an Adaptive MPC: Automated Insulin Delivery in Diabetes. In 2020 European Control Conference (ECC) (pp. 283-288). IEEE.

[70] Aswani, A., Gonzalez, H., Sastry, S. S., & Tomlin, C. (2013). Provably safe and robust learning-based model predictive control. Automatica, 49(5), 1216-1226.

[71] Road Safety Authority, (2011). The two-second rule, Government of Ireland

[72] Yimer, T. H., Wen, C., Yu, X., & Jiang, C. (2020). A Study of the Minimum Safe Distance between Human Driven and Driverless Cars Using Safe Distance Model. arXiv preprint arXiv:2006.07022.

[73] Anon, B. (1997). Mechanical vibration and shock-evaluation of human exposure to whole-body vibration. International Organization for Standardization, ISO, 2631.

[74] R. Sukthankar, “Situation awareness for tactical driving”, Doctoral dissertation, Carnegie Mellon University, 1997.

[75] A. Rolnick en R. E. Lubow, “Why is the driver rarely motion sick?” The role of controllability in motion sickness, Ergonomics, vol 34, no 7, bll 867–879, Jul 1991.

[76] G. BOOK, “CONCEPTS AND RATIONALE FOR STREAMING SERVICES OVER BUNDLE PROTOCOL”, 2018.

[77] D. R. Merrit, “Safe Speeds on Curves: A Historical Perspective of the Ball Bank Indictor”. ITE Journal, Vol. 58, No. 9, 1988, pp. 15–19.

[78] F., E., and F. Navin, “Automobiles on horizontal curves: experiments and nobservations”, Transportation Research Record 1628, no. 1 (1998): 50-56.

[79] H. Caesar et al., “nuScenes: A multimodal dataset for autonomous driving,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 11621–11631

[80] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[81] Fathi, A., & Krumm, J. (2010, September). Detecting road intersections from GPS traces. In International conference on geographic information science (pp. 56-69). Springer, Berlin, Heidelberg.

[82] Lassarre, S., Bonnet, E., Bodin, F., Papadimitriou, E., Yannis, G., & Golias, J. (2012). A GIS-based methodology for identifying pedestrians’ crossing patterns. Computers, Environment and Urban Systems, 36(4), 321-330.

[83] Wang, M., Zhao, Y., & Zhang, B. (2015). Efficient test and visualization of multi-set intersections. Scientific reports, 5(1), 1-12.

[84] Zhou, Y., Fang, Z., Thill, J. C., Li, Q., & Li, Y. (2015). Functionally critical locations in an urban transportation network: Identification and space–time analysis using taxi trajectories. Computers, environment and urban systems, 52, 34-47.

[85] Bruntrup, R., Edelkamp, S., Jabbar, S., & Scholz, B. (2005, September). Incremental map generation with GPS traces. In Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005. (pp. 574-579). IEEE.

[86] Edelkamp, S., & Schrödl, S. (2003). Route planning and map inference with global positioning traces. In Computer science in perspective (pp. 128-151). Springer, Berlin, Heidelberg.

[87] Yanagisawa, Y., Akahani, J. I., & Satoh, T. (2003, January). Shape-based similarity query for trajectory of mobile objects. In International Conference on Mobile Data Management (pp. 63-77). Springer, Berlin, Heidelberg.

[88] Schroedl, S., Wagstaff, K., Rogers, S., Langley, P., & Wilson, C. (2004). Mining GPS traces for map refinement. Data mining and knowledge Discovery, 9(1), 59-87.

[89] Hwang, J. R., Kang, H. Y., & Li, K. J. (2005, October). Spatio-temporal similarity analysis between trajectories on road networks. In International

Conference on Conceptual Modeling (pp. 280-289). Springer, Berlin, Heidelberg.

[90] Chang, M. F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., ... & Hays, J. (2019). Argoverse: 3d tracking and forecasting with rich maps. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8748-8757).

[91] Salzmann, T., Ivanovic, B., Chakravarty, P., & Pavone, M. (2020, August). Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In European Conference on Computer Vision (pp. 683-700). Springer, Cham.

[92] Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., & Urtasun, R. (2020, August). Learning lane graph representations for motion forecasting. In European Conference on Computer Vision (pp. 541-556). Springer, Cham.

[93] Ivanovic, B., & Pavone, M. (2019). The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 2375-2384).

[94] Hegde, V., & Zadeh, R. (2016). Fusionnet: 3d object classification using multiple data representations. arXiv preprint arXiv:1607.05695.

[95] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2117-2125).

[96] Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.

[97] Kessler, T., Dorian, G., & Mack, J. H. (2017, October). Application of a rectified linear unit (ReLU) based artificial neural network to cetane number predictions. In Internal Combustion Engine Division Fall Technical Conference (Vol. 58318, p. V001T02A006). American Society of Mechanical Engineers.

[98] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platin-sky, W. Jiang, and V. Shet. Lyft Level 5 AV Dataset 2019. https://level5.lyft.com/dataset/, 2019. 2, 3

[99] Girgis, R., Golemo, F., Codevilla, F., D'Souza, J. A., Weiss, M., Kahou, S. E., ... & Pal, C. (2021). Autobots: Latent variable sequential set transformers. arXiv preprint arXiv:2104.00563.

[100] Wu, Y. H., Liu, Y., Zhan, X., & Cheng, M. M. (2021). P2T: Pyramid pooling transformer for scene understanding. arXiv preprint arXiv:2106.12011.

[101] Deo, N., Wolff, E., & Beijbom, O. (2022, January). Multimodal trajectory prediction conditioned on lane-graph traversals. In Conference on Robot Learning (pp. 203-212). PMLR.

[102] L. Guo, "Estimating time-varying parameters by the Kalman filter based algorithm: stability and convergence," in IEEE Transactions on Automatic Control, vol. 35, no. 2, pp. 141-147, Feb. 1990, doi: 10.1109/9.45169.

[103] Fitzpatrick, K., & Kahl, K. (1992). A historical and literature review of horizontal curve design.

[104] C. Tay, “Analysis of Dynamic Scenes: Application to Driving Assistance,” Theses, Institut National Polytechnique de Grenoble - INPG, Sept. 2009.

[105] G. Aoude, B. Luders, J. Joseph, N. Roy, and J. How, “Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns,” Autonomous Robots, vol. 35, no. 1, pp. 51–76, 2013

[106] Kim, S., Jeon, H., Choi, J., & Kum, D. (2022). Improving diversity of multiple trajectory prediction based on map-adaptive lane loss. arXiv preprint arXiv:2206.08641.

[107] Niedoba, M., Cui, H., Luo, K., Hegde, D., Chou, F. C., & Djuric, N. (2019, December). Improving movement prediction of traffic actors using off-road loss and bias mitigation. In Workshop on’Machine Learning for Autonomous Driving’at Conference on Neural Information Processing Systems.