UNIVERSITY OF MISKOLC

Intelligent Solutions for Autonomous Vehicle Driving System Optimization

by

Ahmad REDA

Supervised by Dr. József VÁSÁREHELYI

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
József Hatvany Doctoral School for Computer Science and Engineering
Faculty of Mechanical Engineering and Informatics
Institute of Automation and Infocommunication

August 2023

This research was supported by the European Union and the Hungarian State, co-financed by the European Regional Development Fund in the framework of the GINOP-2.3.4-15-2016-00004 project, aimed to promote the cooperation between the higher education and the industry.

Contents

Chapter 1

Introduction

Developing autonomous vehicles is a highly important topic in automotive industry and the field of intelligent transportation systems. A variety of classical control strategies have already proved their merits in this field. However, with the increase in the non-linearity and complexity of the driving system's environment, the efficiency of these approaches drop off due to the limitations of their computing capabilities with such highly complex systems or to lack of efficacy related to maintaining the balance between driving performance and driving smoothness. For example, model predictive control (MPC) is very well known classical control strategy that is used for steering system due to its abilities in solving the optimization problem in real time, handling the system constraints and dealing with changing dynamics of the vehicle. However, its efficiency is negatively affected in high complex environment and may not be able to meet the real time requirements due to the fact that it solves the optimization problem at each time step which massively increases the computational loads. As a result, developing robust systems becoming more crucial and remains an open challenge for researchers and automotive companies alike. The motivation in this work is to contribute to the optimization of the autonomous vehicle driving system. we tackled this problem in two different aspects. In the first part, we focused on optimizing the the implementations of the classical control, precisely MPC control on limited resources platform (low-end FPGA). It is certainly noticeable that the use of artificial intelligence (AI) in this field is unavoidable due to the efficiency that has been achieved in different fields. In the second part we focused on taking advantage of machine learning algorithms to provide an efficient alternative solutions to the classical control. In addition to optimize the deployment of DNN on FPGA using a new innovative tool.

Autonomous vehicles have been researched since 1980, for the time, these researches represented impressive technological advancements. In the 1980s, the Defense Advanced Research Projects Agency put its first 600-meter-distance prototypes on the road. In 2004, the same agency introduced the "DARPA Challenge," which encouraged institutions to innovate in this area. The objective was traveling 240 kilometers through the Mojave Desert without human assistance. Over time, more innovations have gained popularity, and diverse companies and research centers have taken up the challenge of developing a fully autonomous vehicle. Tesla, Waymo, Zoox and automakers like Mercedes and Ford are currently the most well-known in this field [BGC+21], [TMD+06].

The automotive industry has agreed on a definition of autonomous driving, and it is best summarized as" the ability of the vehicle to drive partly or fully without or with limited human interaction". According to the Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems "SAE-J3016", vehicle autonomy is divided into six different levels, from fully manual (level 0) to fully autonomous (level 5). Level 0 (No Automation) depends on the human driver to perform all the driving tasks, it is manually controlled. Level 1 (Driver Assistance) is considered as the lowest automation level, where the driver has full responsibility, but some assistant driving systems are included for certain circumstances. Level 2 (Partial Automation) combines different automated functions which can be working simultaneously, such as steering and acceleration tasks, but the driver is still involved in the driving tasks such as performing the maneuvers and has to monitor the environment all the time. At Level 3 (Conditional Automation) the vehicle has the capability of detecting the surrounding environment and making decisions in normal conditions, but the necessity of the driver still exists, meaning that the driver has to be ready to take control over the vehicle at any time. At Level 4 (High Automation) the vehicle performs all the driving tasks in most circumstances, and the driver still has the option to take control. At Level 5 (Full Automation) the vehicle is capable of performing all driving tasks in all circumstances, and the driver has the option to manually override [Com14], [UDoT18]. Generally speaking, the vehicle needs to be able to coordinate and effectively implement functions under three main pillars in order for it to be able to partially or fully drive. Observing the driving surroundings to detect risks and emergency scenarios, then automatically taking action to protect the passengers and eliminate potential collisions. The Advanced Driving Assistance Systems (ADAS), which include functions like driving assistance, collision protection, and emergency breaking keeps track of safety-related issues.

The ADAS technologies are anticipated to advance and play a crucial role in the autonomous driving system optimization. Autonomous vehicle are made up of three main parts, the vehicle, driving software, and hardware, where it depends on sensor, actuators, complex control algorithms, and powerful processors in order to perform its tasks [DTP21]. The core functions of the autonomous vehicle can be categorised into three main categories: perception, planning, and control. The environment perception provides the vehicle with the required information about the surrounding driving environment, including the vehicle's location, the drivable areas, the velocity, etc. Different sensors and tools can be implemented to tackle the perception task, such as using ultrasonic sensors, cameras, LiDARs (Light Detection And Ranging), or even a combination of these (sensor fusion) to decrease the uncertainty of the data. Based on the collected data, the best scenarios are obtained and the required control actions are made in the planning module in order to drive the vehicle efficiently to the desired location. In the control function, the commands are sent to the actuators to put the control strategy into action [BSDD17]. Adaptive behaviour in autonomous vehicles provide the ability to changes their behavior parameters in accordance with their environment, while a the autopilot technology is used to automatically manage a vehicle's operation without any manual control. Essentially, the core of autonomous driving is the development of systems that can automate the function of driving. Despite the fact that autonomous vehicles have been around for a while, the demand to develop one that is completely functional has accelerated in the past decade.

In a scientific point of view, the work presented in this document globally proposes a research, in the context of automated driving and precisely a contribution in automated steering and safety. After the introduction where an overview of the autonomous vehicles is presented, the necessary theoretical and hardware background of the different aspects of the researches including control strategies for path tracking, Re-configurable Computing and Hardware Acceleration, and the recent machine learning-based applications in the field are presented in the second and the third chapters. Then comes to the contribution part which is provided in the next 4 chapters. The contributions in chapter 4 can be summarised in two main points. First, studying, analysing and improving the implementations of the MPC controller for the task of automated driving especially with changing dynamic systems. Second, the use of rapid prototyping method (hardware/software co-design) to deploy the design on FPGA (hardware-embedded system). The suggested solution in chapter 5 is to develop a deep neural network model based on the behavior of the traditional MPC controller so that the DNN model can replace the MPC controller in high complexity driving system environments. Additionally, one of the motivations behind the work is to propose an alternative tool to implement deep neural networks on low-end FPGA. The main contributions in chapter 6 can be summarized in two main points. The first is leveraging the advantages of reinforcement learning and supervised learning by combining them in one control model in such a way that the reinforcement learning-based model optimizes the actions that are taken by the supervised neural work (DNN) model. The second contribution comes in enriching the research on RL algorithms and paving the way to bring RL closer to real-world implementations. The contribution of chapter 7 comes in the orientation of enriching the studies that have been conducted on reinforcement learning method in the frame of safety automated driving in order to validate its efficiency and stability compared to the used classical control approaches. The last chapter summarise the entire work and the provided theses.

Chapter 2

Intelligent Solutions for Autonomous Vehicle Driving System Optimization

2.1 Model Predictive Control for Autonomous Vehicle Steering System and FPGA Implementations

The autonomous vehicle steering system, a multi-input multi-output (MIMO) system, is challenging to design using traditional controllers due to the interaction between inputs and outputs. Designing a larger system increases the controller parameters requiring tuning. Model Predictive Control (MPC) overcomes this problem, as it is a multi-variable control method taking into account the interactions of the variables in the target system. Achieving a high safety level is also critical for autonomous vehicle systems. This can be provided by an MPC controller, which can handle constraints such as maintaining a safe distance from other cars. The wider applicability of the Model Predictive Controller calls for more efficient hardware architectures for implementation. The aim of this work is to achieve optimal implementation of the MPC controller by increasing the computational speed in order to reduce execution time for optimization. This chapter discussed the implementations of model predictive controller for autonomous vehicle steering task. To deal with changing dynamics systems, a linearized function was used to adapt to the new changes and provide accurate prediction at each time step (Adaptive MPC). One of the most effective solutions, in order to achieve MPC implementations for embedded system applications that have constraints related to the computational time, is the use of hardware acceleration. In this context, the deployments of an embedded MPC controller can be achieved using reconfigurable hardware such as Field Programmable Gate Array or System on Chip, which is popular due to its high computational capabilities, parallel processing and development framework [LYLM09]. In this context, the main contributions of this work can be summarised in two main points. First, studying, analysing and improving the implementations of the MPC controller for the task of automated driving especially with changing dynamic systems. Second, the use of rapid prototyping method (hardware/software co-design) to deploy the design on FPGA (hardware-embedded system). The research applied functional on-target rapid prototyping using Embedded Coder and HDL coder. The suggested implementation method is based on taking the optimization problem of the control method through MATLAB Simulink, Fixed-Point Designer, Embedded Coder, and HDL Coder. The suggested method allows the authors to focus on the verification, validation, and test of the embedded system rather than programming, which in turn gives the ability to refine the design, tune the MPC controller parameters and see the results in the real time. Figure 2.1 presents the MPC controller model that is designed for linear systems (constant dynamics system), while figure 2.2 presents the Adaptive MPC controller model that is designed for nonlinear systems (changing dynamics system) with the Update Plant Model block.

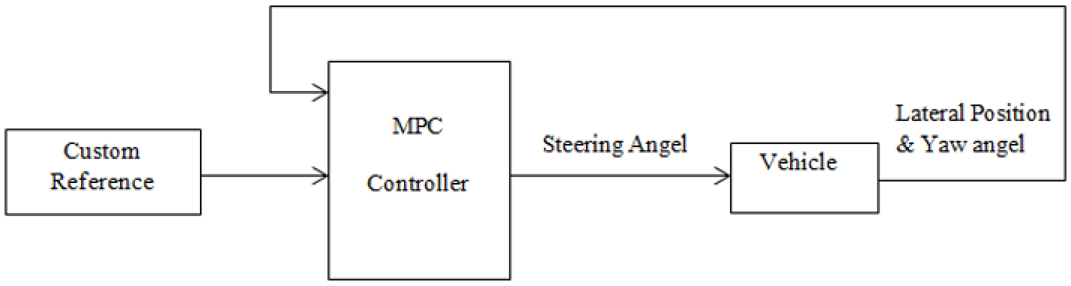

Figure 2.1: MPC controller model for linear system (Constant longitudinal velocity)

The steering system was implemented on System on Chip target using embedded coder and HDL coder. The working methodology is presented in figure 2.3.

MPC controller drives the vehicle to the target point along the desired trajectory by controlling the lateral deviation d and the relative yaw angle θ of the vehicle. Maintaining these variables to be zero or as close as possible to zero is the online optimization problem that the MPC controller must handle in real time. Since MPC is a model-based controller, the design process has two main steps, designing the plant model (the vehicle) first and then designing the MPC controller in the

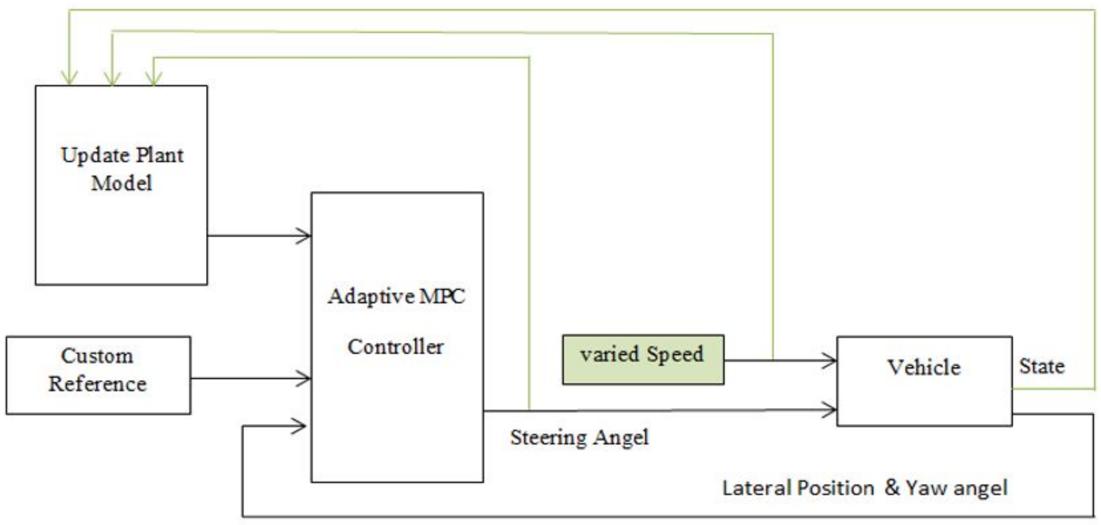

FIGURE 2.2: Adaptive MPC controller model for nonlinear system (varied longitudinal velocity)

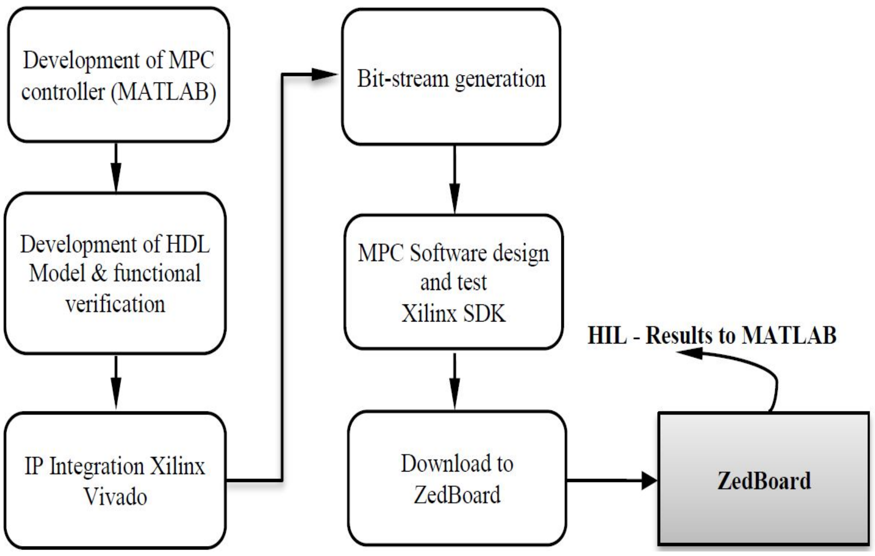
Development of MPC controller (MATLAB)
Development of HDL Model & functional verification
IP Integration Xilinx Vivado
Bit-stream generation
MPC Software design and test Xilinx SDK
Download to ZedBoard
HIL - Results to MATLAB
ZedBoard

FIGURE 2.3: The design workflow of the proposed solution

second step. The design process includes tuning the parameters of the controller and formulating the operating conditions that are imposed by the system in the form of soft and hard constraints.

The implementations and testing of the designed modes (MPC and Adaptive MPC) using MATLAB Simulink show that, MPC successfully drive the vehicle along the desired trajectory in the case of constant dynamics (constant longitudinal velocity), while it failed to handle the system with changing longitudinal velocity. On the other hand, Results demonstrate that using the adaptive MPC controller for the changing dynamics system yields good performance in terms of tracking the reference trajectory.

Both models (MPC and Adaptive MPC controller) were implemented on FPGA and the results were compared with the results obtained using MATLAB Simulink. The experiments showed slight differences in terms of performance between the implementations (Simulink and FPGA). The implementations of MPC and Adaptive MPC controllers on FPGA were analyzed also in terms of resource utilization and power consumption. The results shows that 91% of the total power was used by the Processing System (PS), whereas only 9% was used by Programmable logic (PL).

2.2 Deep Learning-Based Control Strategy for Automated Driving and FPGA Deployments Using a Novel Automatic IP Generator Tool

This chapter concerns the inefficiency of the classical MPC controller with the complex automated driving environment. A deep neural network model-based model was suggested as an efficient alternative to the classical MPC controller. The DNN model was trained to imitate the behaviour of the MPC controller. The designing, testing and validation processes were discussed in details. Also, this chapter concerns the deployment of the DNN model on low end FPGAs, where a new tool based on the Xilinx System Generator was developed to perform and optimize the deployments of the DNN model on FPGAs. Figure 2.4 shows the overall design of the MPC, the plant model and the inputs/outputs signals.

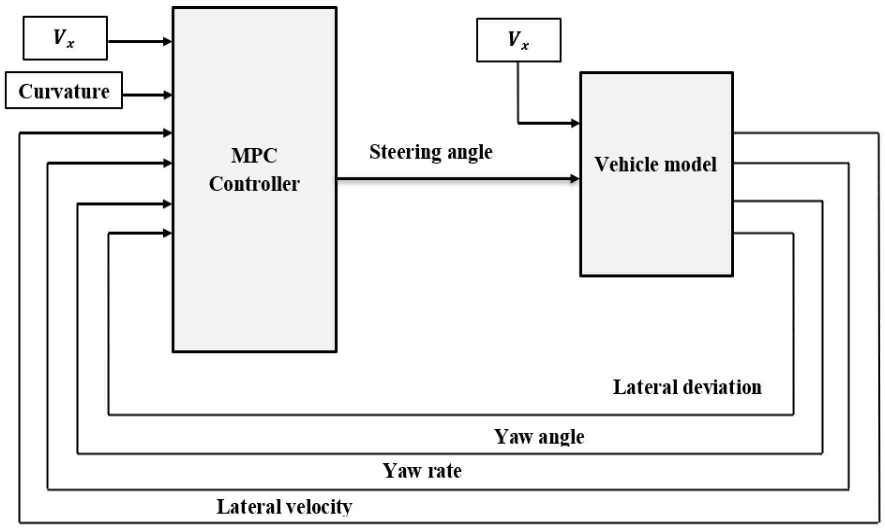

FIGURE 2.4: General MPC and plant model diagram.

After designing, implementing and testing the MPC controller, the suggested DNN model is designed. Designing the deep neural model goes through several steps including designing the model architecture, defining the training options, and preparing the training data. The DNN model is developed using Matlab environment. The suggested architecture of the DNN controller consists of 8 layers which are the input layer, 6 fully connected layers (FC) and each FC layer has activation function (ReLU). The output layer is a regression layer which holds the loss function (mean-squared-error). Six observations are determined as inputs: yaw angle (θ), lateral velocity (v_y), lateral deviation (d), yaw rate (ω), the curvature (ρ), and the previous control action ($\hat{\delta}$). The steering angle δ is the output (control action). The detailed structure is shown in figure 2.5.

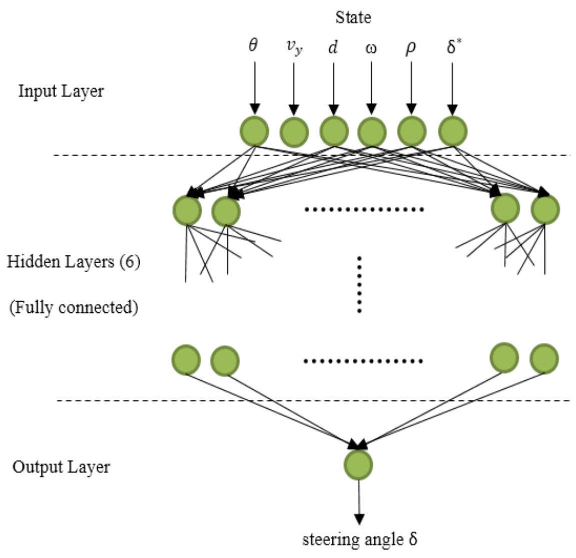

FIGURE 2.5: The DNN model architecture representation.

The trained neural network was tested using the testing data set. The performance of the DNN model is evaluated comparing to the performance of the MPC controller, where the RMSE between the outputs of the controllers is calculated. The obtained root mean square error by the end of the testing process was: RMSE = 0.011228, which is a very small compared to the range of the steering angle [-60 – 60]°. This small value indicates to that the DNN model successfully imitated the behavior of the MPC controller.

The procedure of the neural network's IP auto-generating has two main steps, as shown in figure 2.6. First, the user is asked to define the parameters concerning the DNN (the structure, the data type and the activation function) and the targeted computational HW and the values of weights and biases parameters are imported from the pre-trained DNN. Then comes the step of setting the input/output interfacing mode. After setting up the DNN-IP preferences, the XSG automation part begins, which consists of invoking the elementary computational components needed for each neuron, linking the components and the neurons, setting the weights and biases accordingly, implementing the I/O, and then generating the IP. Figure 2.7 shows the auto-generated DNN circuit on the Xilinx System Generator to be implemented on a low-end FPGA.

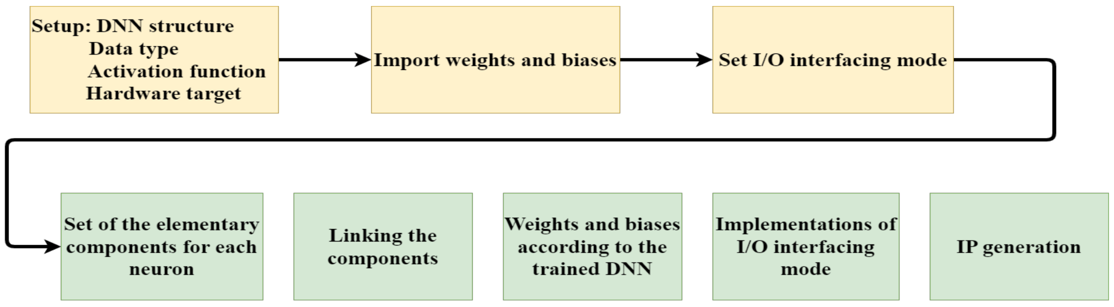
Setup: DNN structure
Data type
Activation function
Hardware target
Import weights and biases
Set I/O interfacing mode
Set of the elementary components for each neuron
Linking the components
Weights and biases according to the trained DNN
Implementations of I/O interfacing mode
IP generation

FIGURE 2.6: Flow of deep neural network IP automated generating.

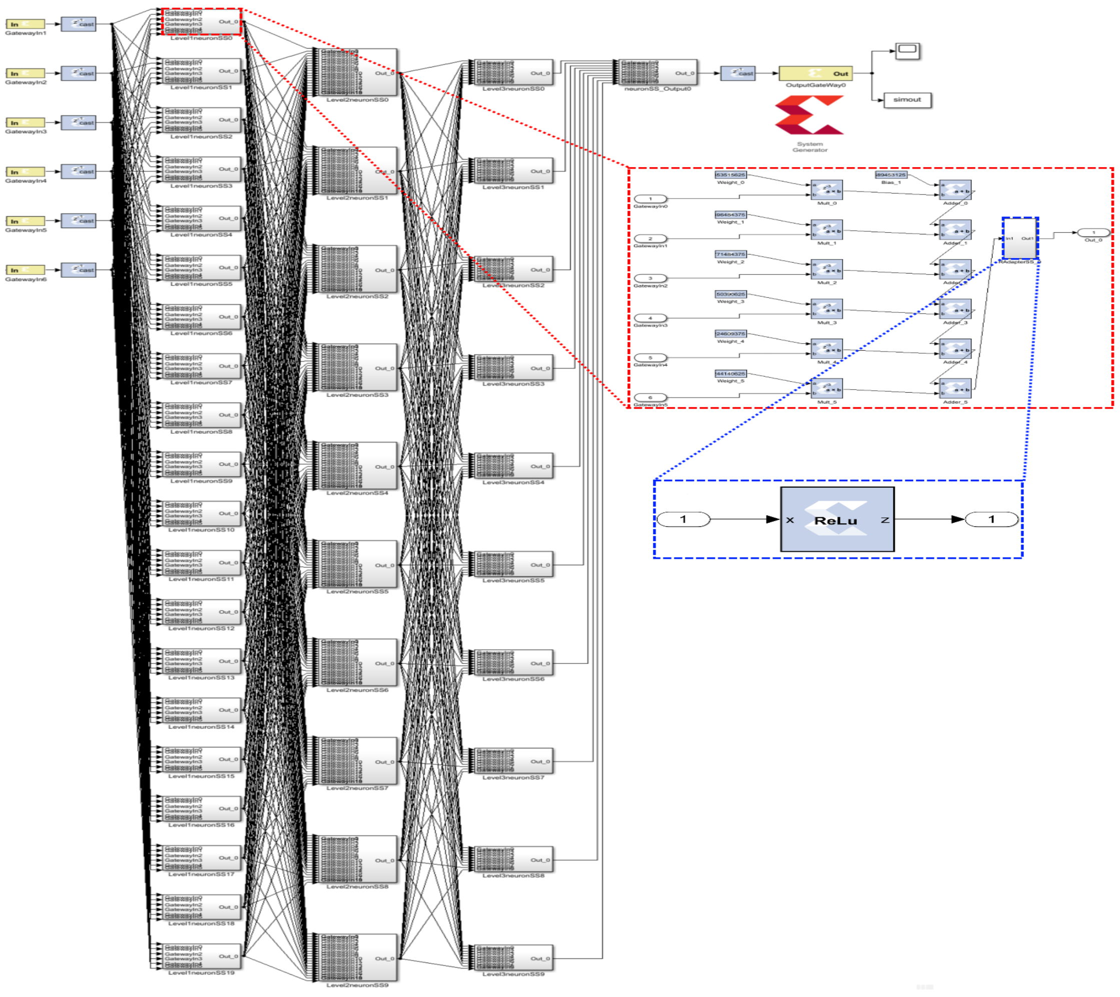

FIGURE 2.7: The auto-generated deep neural network structure on Xilinx System Generator to be implemented on a low-end FPGA.

Figure 2.8 shows the detailed steps of "Automatic DNN IP Generator" implementation.

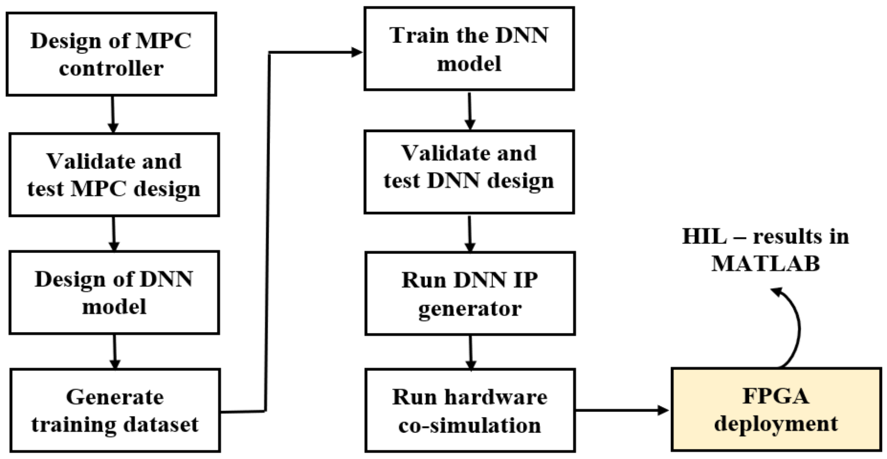

FIGURE 2.8: The implementation steps of the solution.

The obtained results show that the suggested DNN model successfully and efficiently imitates the behavior of the classical MPC controller and reduced the execution time by 3 to 4 times. The trained DNN and the MPC controller behave similarly with very small output deviation, and the maximum difference is approximately 0.0094 rad (0.53 degrees). Both controllers were able to follow the desired trajectory by driving the lateral deviation and yaw angle to be very close to zero. Additionally, and taking into consideration the control system characteristics, the results clearly show that both controllers were able to reach the stable state at almost the same time with the same amount of overshooting. As a result, the traditional model predictive controller can be replaced efficiently by the suggested DNN model. On the other hand, the trained DNN model was efficiently deployed on low-end FPGA *Xilinx Kintex-7 FPGA KC705* using floating and fixed-point data type, achieving satisfactory performance and meeting the design's constraints.

2.3 A Hybrid Machine Learning-Based Control Strategy for Automated Driving Optimization

This chapter concerns the use of machine-learning algorithms for automated driving, where a new method is introduced to achieve performance optimization. The new method leverages the advantages of supervised learning and reinforcement learning algorithms in one control model in such a way that the reinforcement learning model optimizes the actions that are taken by the supervised neural work (DNN). To sum up, three different machine learning-based models were developed to perform an autonomous driving task: a supervised learning model (deep neural network - see chapter 5), a reinforcement Deep Q-learning model (DQN), and the hybrid model. The DQN was structured similarly to the DNN and trained by directly interacting with the driving environment. The hybrid model is a combination of supervised and reinforcement learning algorithms. The behaviors of the suggested models were compared based on several performance indicators, including the ability to drive the vehicle along the desired trajectory, the response time, and the smoothness of the driving system. The combined method is expected to provide an optimized solution, as the actions that are taken by the decision maker (trained DNN) will be evaluated and optimized by another neural network in order to minimize errors. Additionally, the combined model will be able to deal with and adapt to new cases that have not been faced during training. The desired models are designed taking into consideration the same dynamics of the vehicle, the constraints, and the environment conditions that were used to design the DNN model in chapter 5. The designing processes went through several steps, preparing the environment, creating and training the agents, and finally testing and evaluating the behaviours. The suggested hybrid model is designed in a way that the trained DNN model is used as a decision-maker (Actor) in a Deep Deterministic Policy Gradient (DDPG) reinforcement learning model. To create the agent, besides having the trained DNN model as an actor, the critic is created based on the actions-observations specifications, where its neural network is structured to accept seven inputs (state-action) and one outputs (the corresponding expected long-term reward $Q(s, a \mid \Phi^Q)$), and 3 hidden layers. Figure 2.9 shows the detailed structure of the combined model.

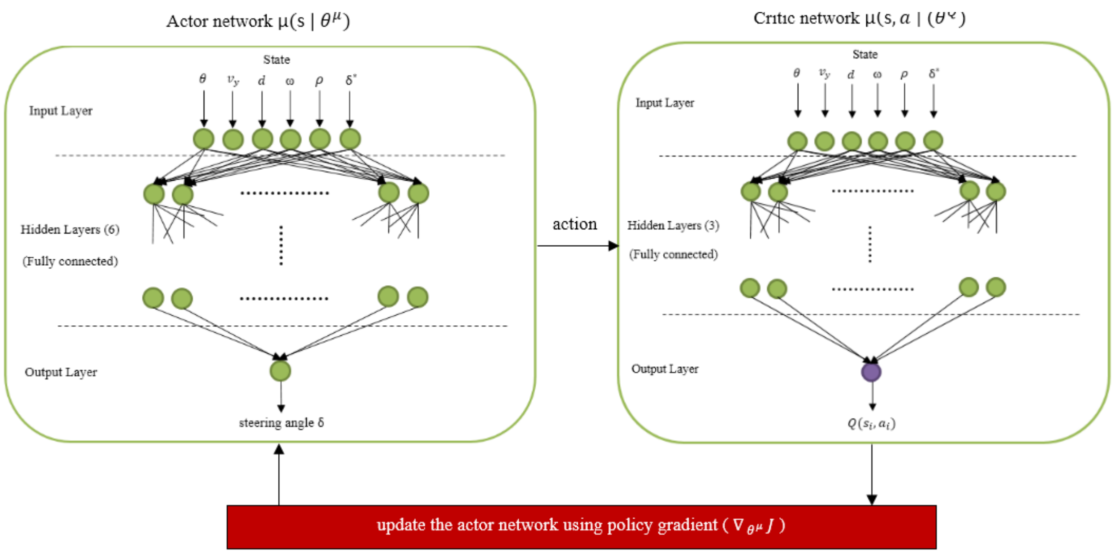
Actor network μ(s | θ^μ)
Critic network μ(s, a | (θ^Q)
State
θ v_y d ω ρ δ*
Input Layer
Hidden Layers (6)
(Fully connected)
Hidden Layers (3)
(Fully connected)
Output Layer
steering angle δ
$Q(s_t, a_t)$
action
update the actor network using policy gradient ($\nabla_{\theta^\mu} J$)

FIGURE 2.9: The structure of the actor-critic networks – combined model

On the other hand, the suggested DQN model is structured and designed to accept the state from the environment as an input (vector with 6 observations) and outputs the estimated Q-value of each possible discrete action that can be taken at that state (vector of n=121 Q values). The detailed structure of the DQN model is shown in figure 2.10.

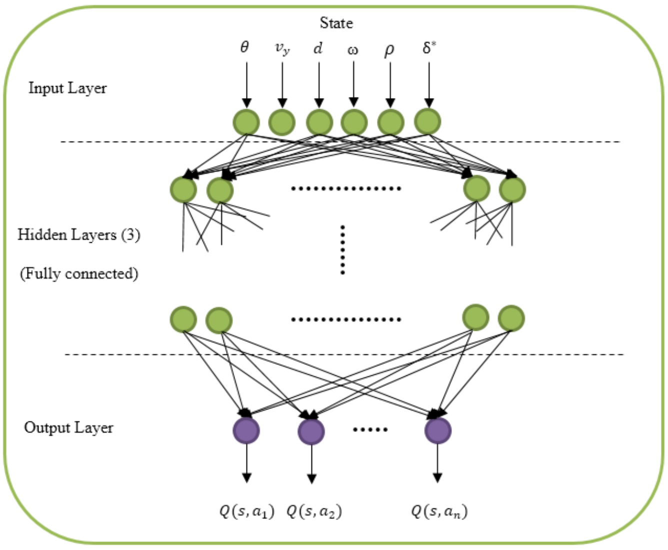

FIGURE 2.10: The reinforcement DQN structure

The performances of the suggested hybrid (RL-supervised) model is compared to the DNN and DQN models to evaluate the achieved improvements compared to the other machine-learning-based algorithms. The obtained results show that the hybrid model responded in a way that improved the smoothness of the driving system by reducing the over shootings. It drove the lateral deviation to be very close to zero (0.003 m) in a reasonable time, compared to the DNN model which achieved 0.0009 m as a final value at almost the same time but with higher overshooting and thus higher lateral deviations. The DQN model was not as efficient as the other models, where its behavior led to higher overshooting and drove the lateral deviation to a final value of 0.01 m. As a result, and taking all the performance indicators into consideration, one can state that the combined model provided the best result and achieved the expected optimization by driving the vehicle to the reference target more smoothly and within a reasonable time. This work shows the efficiency of combining supervised and reinforcement learning to leverage the advantages of both algorithms, where supervised learning speeds up the learning process and reinforcement learning improves self-adaptation to new states, which in turns makes it more efficient within the complex driving environment

2.4 Contribution to Automated Driving Safety Using Reinforcement Learning and FPGA Deployments

This chapter concerns the necessity of developing an alternative control approach for safe automated driving that has the capability to deal efficiently with the complexity, non-linearity and uncertainty of vehicle dynamics. Even with the successful implementations of the reinforcement learning in real world for different applications, it is still not commonly used compared to the supervised and unsupervised learning. In this work, a reinforcement learning-based framework is suggested as an alternative solution to the classical control for the application of maintaining a safe distance in autonomous driving system. The efficiency and stability of the suggested model is evaluated compared to the very well known model-based control (MPC) which was developed and implemented for the same task and under the same conditions and constraints. Additionally, in order to verify the efficiency of the suggested model in practice, the trained RL model was deployed and tested on low end FPGA-in-the-loop. MPC and RL -based controllers are designed to respond to environmental changes using two control modes, speed and maintain modes. In the case where the relative distance (d_{rel}) between the two vehicles (ego vehicle and leading vehicle) is greater than the reference safety distance (d_{saf}), the controller applies the speed mode that makes the ego vehicle drive at the reference velocity. In the case where the relative distance is less than the safety distance, the controller switches to the maintain mode, and the vehicle drives at the speed of the leading vehicle to keep a safe distance (figure 2.11).

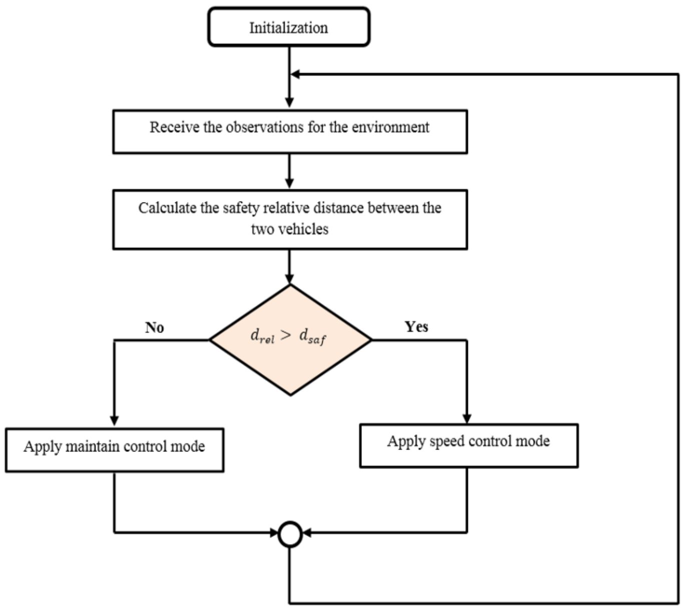
Initialization
Receive the observations for the environment
Calculate the safety relative distance between the two vehicles
$d_{rel} > d_{saf}$
No
Yes
Apply maintain control mode
Apply speed control mode

FIGURE 2.11: The schema of applying the control modes.

Designing the MPC controller went through several steps, including vehicle model design, determining the input/output signals and the design parameters of the control system as it is shown in figure 2.12 and table 2.1. The measured outputs are used for state estimation, while the manipulated variables are the optimal control actions. The MPC design parameters and the control constraint are presented in table 2.2. Figure 2.13 shows the overall workflow of the MPC-based control system.

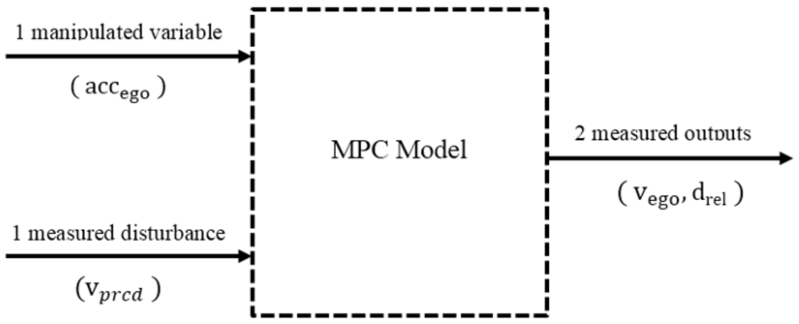

FIGURE 2.12: MPC plant model- Inputs/outputs.

TABLE 2.1: Inputs/Outputs signals of MPC controller.

Signal type	Parameter	unit	Description
Measured outputs (MO)	v_{ego}	m/s	Longitudinal velocity of the ego vehicle
	d_{ref}	m	The relative distance between the proceeding and the ego vehicles
Measured disturbance (MD)	v_{prcd}	m/s	Longitudinal velocity of the proceeding vehicle
Manipulated variable (MV)	acc_{ego}	m/s^2	acceleration\deceleration
References	v_{re}	m/s	Reference velocity in speed mode
	d_{saf}	m	The reference safety distance

TABLE 2.2: MPC design parameters.

MPC controller parameters	
Sample time (T_s)	0.1 s
Prediction horizon (P)	30
Control horizon (M)	3
Control action constraints	
Acceleration	[-3, 3] m/s^2

In this work, the Deep Deterministic Policy Gradient algorithm is used to design the reinforcement leaning controller. This algorithm uses two different deep learning-based approximators, the actor and the critic. The design process went

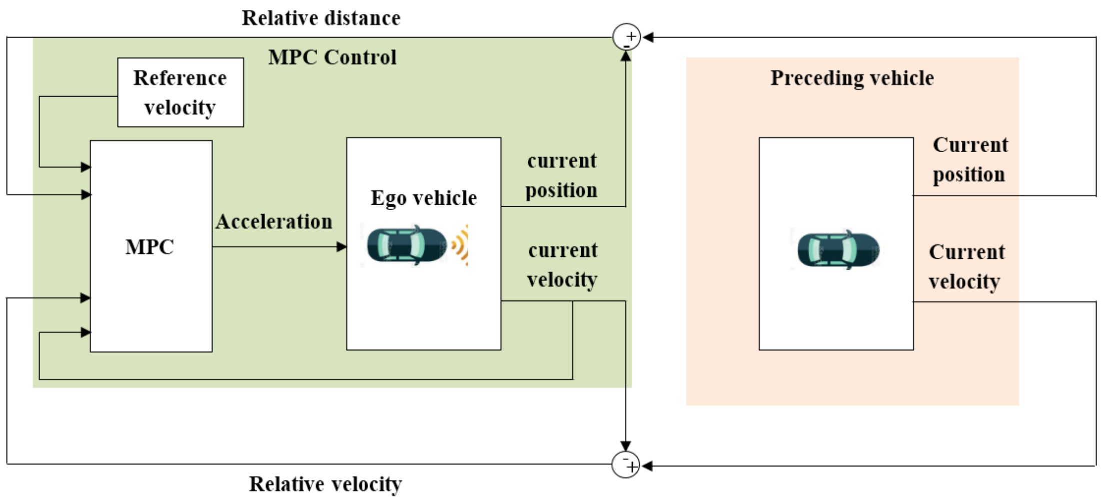

FIGURE 2.13: Overall diagram of the control system-MPC model.

through several steps, starting from preparing the environment, designing the neural networks of the actor and critic, creating and training the agent, and finally running, testing and validating the model. The observations of the environment are determined to be the vehicle velocity (v_{ego}), the velocity error (v_{er}) and the integral velocity error (p_{er}). Velocity error represents the difference between the reference and the vehicle velocity. The acceleration constraint is determined to be in the same range as that of the MPC controller, [-3, 3] m/s^2. Figure 2.14 shows the neural network structure of the actor-critic approximators. Figure 2.15 shows the overall workflow of the RL-based control system.

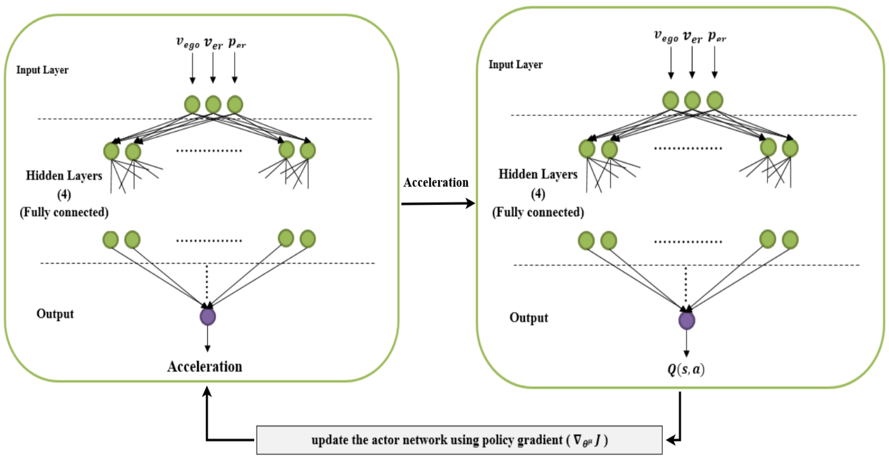
v_{ego} v_{er} p_{er}
Input Layer
Hidden Layers (4) (Fully connected)
Output
Acceleration
Acceleration
v_{ego} v_{er} p_{er}
Input Layer
Hidden Layers (4) (Fully connected)
Output
$Q(s,a)$
update the actor network using policy gradient ($\nabla_{\theta^{\mu}} J$)

FIGURE 2.14: RL model: actor – critic neural network structure.

Due to the fact that the reinforcement learning algorithm depends on trial and error, it is safer to test the design using the MATLAB Simulink first, before the implementation on the real SOC. For this purpose, multiple simulations were performed and the final trained policy that meets the requirements is taken to the

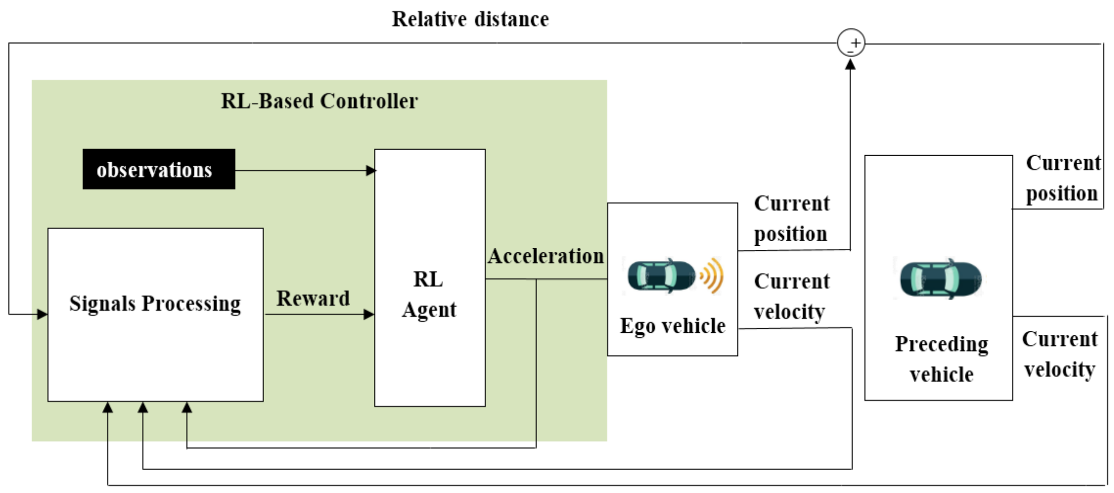

FIGURE 2.15: Overall diagram of the control system-RL model.

next step to be deployed on the target FPGA. The suggested RL-based control model is deployed on a low-end SOC (ZedBoard). The deployment process went through several steps (see figure 2.16). In the first step, the trained policy is extracted from the trained agent (the Simulink model) and represented in C code generated using MATLAB Embedded Coder. The generated code accepts the environment observations as inputs, and outputs the optimal action for the current state based on the trained policy. In the next step, the hardware configurations were prepared in order to perform the communication between SOC and MATLAB Simulink. The deployment is performed by downloading and running the generated code on the target SOC. Running the model on Zedboard goes through the following cycle: SOC receives the signals from MATLAB Simulink through the communication channel, executes the algorithm, outputs a control action (acceleration or deceleration), and then sends it to the Simulink model to update the state of the driving environment.

The results show that both controllers responded efficiently to the environment's changes based on the control modes and the design specifications. In detail, at the beginning the RL controller drove the vehicle from the initial state to the reference velocity in approximately 4.5 s, while it took around 6.2 s in the case of MPC controller. The results also show a stable behaviour of the RL model which responded faster to the environment changes compared to the MPC controller. RL-based model improved the response time with 1.75 s in average. On the other hand, the behavior of the RL controller shows a slight higher overshooting in terms of following the reference speed especially against the initial state, where the maximum overshooting was approximately 1.3 m/s. As a conclusion, the results demonstrate the advantage of the reinforcement learning model in term

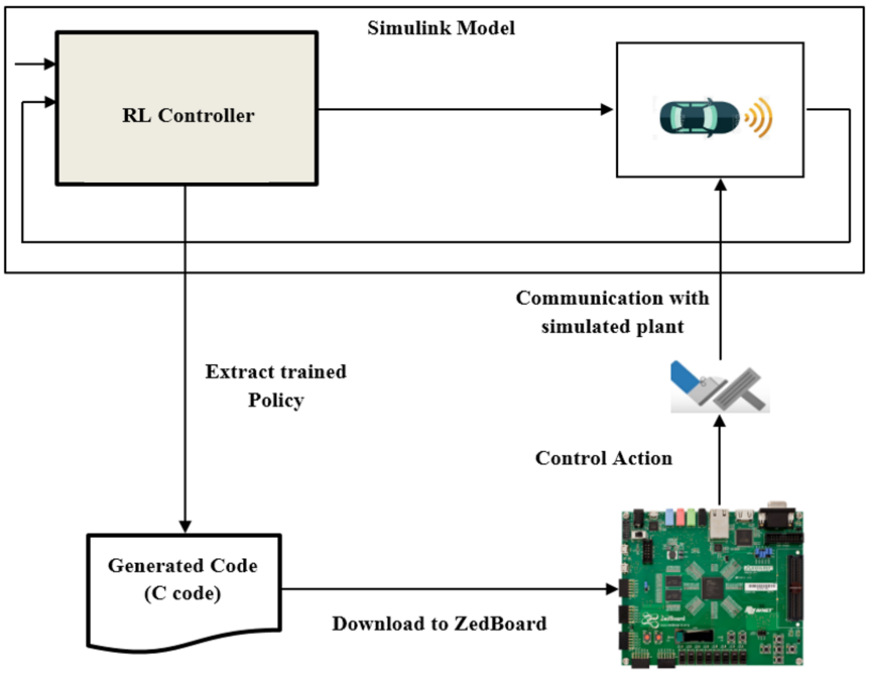

Figure 2.16: RL training algorithm.

of the ability to predict and follow the changes in the environment state faster than the MPC controller. This result is compatible with the fact that solving the optimization problem using a trained neural network is faster than solving an online quadratic problem at each time step (MPC controller).

Chapter 3

Summary

Given the great importance of the autonomous vehicles in reducing the traffic risks and accidents on the roads resulting from human errors, engineers in the automobile sector have worked extremely hard over the past decade to develop and implement control strategies in an effort to improve road transportation and reach a fully autonomous and safe vehicles. Without a doubt, the next few years will witness a significant increase in fully autonomous vehicles on the roads. The main motivation of my doctoral work is to contribute to this efforts by making use of the advanced technologies and algorithms to achieve the desired optimization. This contributions can be summarised into four main parts. It is known that the efficiency of the classical control strategies drops off in highly complex environments due to their weakness in handling the dynamically changing systems and high processing demands, especially when it comes to the limited resources of embedded computing platforms such as system-on-chip and field-programmable gate array.

In the first part, I addressed these problems for model predictive control as on of the most traditional control strategies that is used for automated steering task. The aim was to optimize the implementations of MPC controller. To deal with dynamic changes systems, the adaptivity concept was used, where the optimization problem remains the same (same number of states and constraints) but a new linear model is used at each time step to obtain an accurate prediction based on the new conditions, and that is called adaptive model predictive control. The obtained results showed that adaptive MPC was able handle the chaining dynamics and yields a good performance in terms of tracking the reference trajectory. In the second section of this work, I applied functional on-target rapid prototyping using embedded coder and HDL coder (hardware/software co-design) for the

implementation of embedded systems dedicated for digital signal processing considering performance, execution time and resources consumption. The suggested implementation method is based on taking the optimization problem of the control method through MATLAB Simulink, Fixed-Point Designer, Embedded Coder and HDL coder, which allows the authors to focus on the verification, the validation and the test of the embedded system rather than programming, which in turn gives the ability to refine the design. Different strategies were implemented to achieve the resource optimization, where the implementations involve Logical optimization, placement of logic cells, and routing the connections between cells.

In the second part, I worked on developing DNN-based controller as alternative to the classical MPC in the frame of automated driving task, suggesting that the use of a deep neural network can significantly increase efficiency and inevitably result in reduce the time and the complexity of implementations. The suggested DNN model, was designed and trained to imitate the behaviour of the traditional MPC. Considering the crucial function that deep neural network hardware implementations play, a new automatic IP generator has been developed in order to deploy and optimize the implementations of the DNN based-models on Field Programmable Gate Array. The performance of the suggested DNN controller after being deployed on low end FPGA was evaluated, and the obtained results show that it is successfully imitated the behaviour of the classical MPC, provided a good performance, and improved the execution time (up to 4 times faster).

In the third part, I suggested a new hybrid machine-learning model that combines two controllers (DNN-based controller and RL-based controller) in one model. The idea behind the hybrid model is to leverage the advantages of the supervised learning and reinforcement learning algorithms in one control model in a way that the RL controller optimizes the actions that are taken by the supervised DNN controller that is developed in chapter 5. The efficiency of the hybrid model was evaluated compared to the supervised DNN and reinforcement DQN models. The obtained results show that the combined model was able to provide the desired optimization by driving the vehicle to the reference speed more smoothly and within a reasonable time. These results proved that the suggested hybrid mode has better generalization capability within the complex driving environment. The supervised learning speeds up the learning process while the reinforcement learning improves self-adaptation to new states that the model was not faced with in the training process.

Although the promising results achieved by the implementations of reinforcement learning algorithms in different field, RL still an emergent field and the real-world applications still very much open challenge and has not yet been applied to practice as successfully as supervised and unsupervised learning. In this context, in the fourth part of this work, I suggested a reinforcement learning-based framework as an alternative solution to the classical control for the application of maintaining a safe distance in autonomous driving system, which enriching the research on RL algorithms and paving the way to bring RL closer to real-world. Additionally, RL model was deployed on a low-end FPGA/SOC after being verified in MATLAB Simulink. In order to perform the deployment, first I extracted the policy from the agent after being trained and I represent it as a C code which was downloaded and tested on SOC. The obtained results showed that the RL model was able to handle the environment changes more efficient compared to MPC controller and improved the response time with 1.75 s in average. Additionally, the method that I used to perform the deployment is evaluated, and the results showed the efficiency of this method, where the RL model provided a stable behaviour after being implemented on FPGA compared to the Simulink behaviour.

Table 3.1 summaries behaviour comparison between the suggested machine learning-based controllers and the traditional MPC in the frame of automated driving task, taking into concentrations the overshooting, settling time, and the execution time as reference indicators. All the controllers were evaluated under the same environment, initial state, conditions, and constraints.

Table 3.1: Summary of the suggested ML-Based controllers compared to traditional MPC for the task of automated driving

Application	Controller	Observation	Overshooting (m)	Settling Time (s)	Execution Time (s)
Automated driving	MPC Controller	Lateral deviation	0.036	0.60	1.54
		Yaw Angle	0.093	1.05	
	DNN-Based Controller	Lateral deviation	0.035	0.61	0.34
		Yaw Angle	0.094	1.03	
	DQN - Based Controller	Lateral deviation	0.182	1.3	0.37
		Yaw Angle	0.082	1.2	
	Combined Controller (DNN-RL)	Lateral deviation	0.0034	0.5	0.23
		Yaw Angle	0.0576	1.3	

3.1 Theses

Thesis I

I gave a methodology for adaptive MPC development for the changing dynamics system, which significantly improved the behaviour of the controller in terms its ability to predict and follow the dynamic changes of the system efficiently after reaching the steady state. Resulting to improve the performance and the smoothness of the driving system. To perform embedded system's deployment, I applied and analyzed different implementations of the proposed method from source utilization point of view. The implementations included logical optimization, placement of logic cells, and routing the connections between cells.

Related Publications: [RBV20], [RV20], [BRV20]

Thesis II

I proposed deep neural network-based control strategy as an alternative solution to the classical MPC controller for automated driving task aiming to reduce the complexity of solving the online-optimization problem, therefore the execution time. I designed and trained the DNN-based controller to imitate the behaviour of the traditional MPC controller. I also proposed a new automatic intellectual property generator tool, which is developed not only to perform but also to optimize the deployments of deep neural networks on low-end Field FPGA.

Related Publications: [RBV21], [KRVB20a], [ch5]

Thesis III

I proposed machine learning-based control strategy that combines supervised learning (DNN model) and reinforcement learning (RL-model) algorithms in one controller, aiming to achieve the optimization by leveraging the advantages of these algorithms in a way that the RL controller optimizes the actions that are taken by the supervised DNN controller. I evaluated the efficiency of the hybrid model compared to the supervised DNN and reinforcement DQN models which are developed for the same task. The combined model provided the best result and achieved the expected optimization by outputting accurate control actions which reduced the overshooting behaviour, resulting a significant improvement in terms of the smoothness of the driving system. **Related Publications:** [ch623], [KRVB20b], [RBV21]

Thesis IV

I proposed reinforcement learning-based control strategy for the task of maintaining a safe distance in the frame of automated driving system. I also proposed a method to deploy the developed model on low-end FPGA. The method extracts the policy of *the trained RL-agent and converts it to a C code that is downloaded and run of the target SoC (FPGA). Compared to the traditional MPC controller, the results show the superiority of the reinforcement learning - based model in term of the ability to predict and follow the changes in the environment state and improvement in response time (1,75 s in average). This work contributed to enriching the research on RL algorithms and paving the way to bring it closer to real-world.*

Related Publications: [RV23]

Bibliography

[BGC$^+$21] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago Paixão, Filipe Mutz, Lucas de Paula Veronese, Thiago Oliveira Santos, and Alberto De Souza. Self-driving cars: A survey. *Expert Systems with Application*, 165(1), 2021.

[BRV20] Rabab Benotsmane, Ahmad Reda, and Jozsef Vasarhelyi. Model predictive control for autonomous quadrotor trajectory tracking. In *2022 23rd International Carpathian Control Conference (ICCC)*, pages 215–220, 2020.

[BSDD17] Andrew Best, Narang Sahil, Barber Daniel, and Manocha Dinesh. Autonovi: Autonomous vehicle planning with dynamic maneuvers and traffic constraints. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2629–2636. IEEE, 2017.

[ch5] Model predictive-based dnn control model for automated steering deployed on fpga using an automatic ip generator tool. (220).

[ch623] A hybrid machine learning-based control strategy for autonomous driving optimization. (1):–, 2023.

[Com14] SAE On-Road Automated Vehicle Standards Committee. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. In *Technical Report*, pages 1–6. 2014.

[DTP21] George Dimitrakopoulos, Aggelos Tsakanikas, and Elias Panagiotopoulos. A path of structural transformation for the automotive and insurance industries toward autonomous vehicles. In *Autonomous Vehicles*, pages 69–83. Elsevier, 2021.

[KRVB20a] Ashraf Kasem, Ahmad Reda, József Vásárhelyi, and Ahmed Bouzid. Fpga-based intelligent solutions for autonomous vehicles: A short survey. In *The 1st Conference on Information Technology and Data Science*, pages 94–96, 2020.

[KRVB20b] Ashraf Kasem, Ahmad Reda, József Vásárhelyi, and Ahmed Bouzid. A survey about intelligent solutions for autonomous vehicles based on fpga. *CARPATHIAN JOURNAL OF ELECTRONIC AND COMPUTER ENGINEERING*, 13(2):7–11, 2020.

[LYLM09] Mark S. K. Lau, S. P. Yue, K. V. Ling, and J. M. Maciejowski. A comparison of interior point and active set methods for fpga implementation of model predictive control. In *2009 European Control Conference (ECC)*, pages 156–161. IEEE, 2009.

[RBV20] Ahmad Reda, Ahmed Bouzid, and József Vásárhelyi. Model predictive control for automated vehicle steering. *ACTA POLYTECHNICA HUNGARICA*, 17(7):163–182, 2020.

[RBV21] Ahmad Reda, Ahmed Bouzid, and Jozsef Vasarhelyi. Deep learning-based automated vehicle steering. In *22nd International Carpathian Control Conference (ICCC 2021)*, pages 1–5, 2021.

[RV20] Ahmad Reda and József Vásárhelyi. Model-based control strategy for autonomous vehicle path tracking task. *ACTA UNIVERSITATIS SAPIENTIAE ELECTRICAL AND MECHANICAL ENGINEERING*, 12(2020):35–45, 2020.

[RV23] Ahmad Reda and József Vásárhelyi. Design and implementation of reinforcement learning for automated driving compared to classical mpc control. *Designs*, 7(1):1–18, 2023.

[TMD$^+$06] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Mark Oakley, Celia Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, , Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[UDoT18] USA U.S Department of Transportation. *Preparing for the Future of Transportation: Automated Vehicles 3.0*. 2018.