MASTER

# Study of Dynamically Reconfigurable Algorithmic Approximation on Quality of Control

Huang, Yingkai

*Award date:*
2020

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Electronic Systems Research Group

# Study of Dynamically Reconfigurable Algorithmic Approximation on Quality of Control

*Master's Thesis*

Yingkai Huang

Supervisors:
Assistant Professor Dr. Dip Goswami
Ph.D. Candidates Sajid Mohamed and Sayandip De

1.0 version

Eindhoven, May 2020

# Abstract

In recent years autonomous driving is becoming a hot topic with the continuous development of deep learning, computer vision and sensor technology. It comes with robustness and quality of control issues when an Image-Based Control system is targeted to edge devices with limited energy, memory and computing resources. Using end-to-end CNN-based approaches can improve robustness, however, will increase runtime and result in a low frame per second. On the other hand, the traditional hardware-efficient approaches are lack of situation-awareness. Different environmental factors (e.g. road layouts, types of lane markers and weather) have a great impact on lane detection accuracy, and thus influence the quality of control. As a result, traditional approaches can not ensure robustness in the real world.

In this work, we propose a hardware- and situation-aware sensing method to a lane-keeping assist system with the traditional lane detection algorithm to make the system both hardware-efficient and robust. We define situations based on different features and identify them using light-weight CNN-based situation classifiers. Depending on the current situation, we dynamically configure the system knobs based on hardware- and situation-aware characterization. To show the effectiveness of our approach, we consider a hardware-in-the-loop framework on NVIDIA AGX Xavier platform. Besides, Webots is used as the simulation environment on the server.

Our results show that the robustness is highly improved comparing to traditional approaches. Moreover, the quality of control is also 32% better due to the approximated image processing signal pipeline and predefined invocation scheme.

# Preface

This thesis is made as a completion of my master education in TU/e. I am grateful to those who have offered me encouragement and support during the course of my study.

First and foremost, my sincere thanks go to Professor Dip Goswami, my supervisor,who has offered me numerous valuable comments and suggestions with incomparable patience and encouraged me profoundly throughout my postgraduate study. Without his painstaking teaching and insightful advice, the completion of this thesis would have been impossible.

Also, I owe many thanks to my project tutor Sayandip De and Sajid Mohamed, for helping me to solve the problems and overcome the research obstacles. Working with these two patient and professional researchers is my fortunate.

Last but not least, I am deeply indebted to my family and friends, who have helped me and shared with me my worries, frustrations, and happiness, especially during the difficult COVID-19 pandemic .

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Introduction

Autonomous driving has gradually become a hot topic since human error caused most traffic accidents [36]. The earliest research on autonomous driving can be traced back to the 1980s such as ALV project funded by the United States' Defense Advanced Research Projects Agency(DRAPA) [16] and Eureka Project PROMETHEUS carried out by Mercedes-Benz and Bundeswehr University Munich [34]. With the continuous development of deep learning [21], computer vision and sensor technology, autonomous driving technology has achieved breakthrough progress in recent years. Various systems, such as advanced driving assistance systems (ADAS) and lane departure warning systems (LDWS) [4], have been equipped in the current automatic vehicles.

Image-Based Control (IBC) system plays an important role in autonomous driving. More cameras are predicted to be used in automated vehicles over the next decade, with a 20% increase in cameras per vehicle by 2023 [33]since they need to send reliable images to IBC systems. A typical IBC system, which is shown in Figure 1.1, consists of a sensing task($T_s$) including an image signal processing (ISP) pipeline and perception(PR), a computation task($T_c$) executing the control algorithm and an actuation task($T_a$) executing instructions issued by the controller. The quality of control performance is highly determined by the sensing time ($T_s$) for image processing. In practice, such a sensing task is commonly mapped to an edge device which has relatively restricted resource. Thus ($T_s$) becomes the bottleneck of the IBC system.

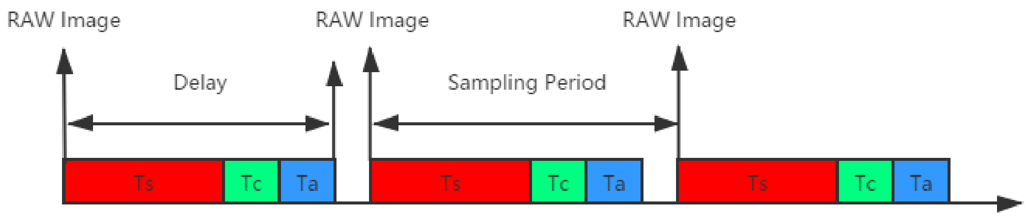

Figure 1.1: Tasks in an IBC system

A lane-keep assist system (LKAS) is considered in this work which is adapted from [9] [26]. Figure 1.2 shows the framework of this closed-loop system. The camera on the car will capture a RAW image every sampling period and send it to the image signal processing (ISP) pipeline. The ISP pipeline will then transform the RAW image to an RGB image. Five stages are considered: demosaicing, denoising, color transform, gamut mapping and tone mapping [5], which will be illustrated in Chapter 3.2. The processed image will be sent to perception (PR), where the lateral deviation of the vehicle from the middle of the lane is calculated. PR will be illustrated in

Chapter 3.3. The lateral deviation is used in the controller to get a steering angle. The controller is designed based on the sampling period and sensor-to-actuation delay, and this will be discussed in Chapter 3.4. Finally, the vehicle will use the steering angle to keep itself in the middle of lane.

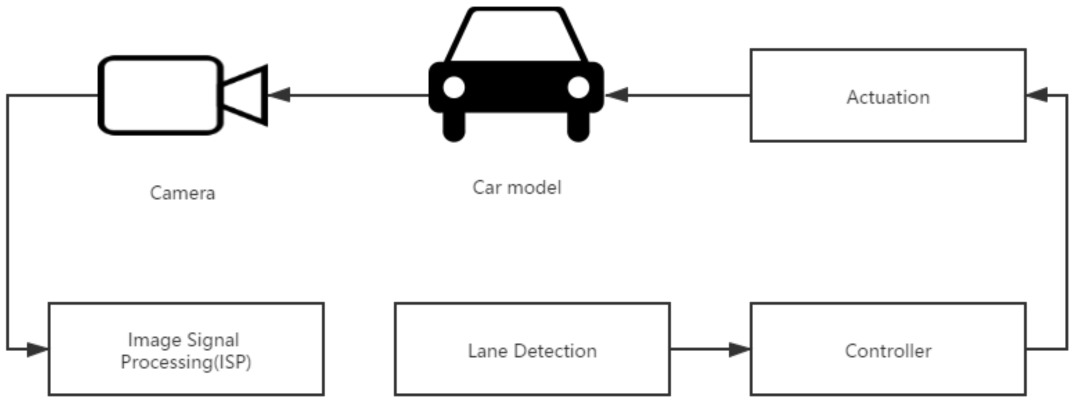

Figure 1.2: Framework of the Considered LKAS

## 1.2 Motivation

To ensure the safety of the autonomous vehicle, the robustness should be given priority when an LKAS is designed. Recently convolutional neural network (CNN) based lane detection technologies such as LaneNet [30] and VPGNet [22] address the robustness. These approaches perform end-to-end learning for different situations, so they can keep the vehicle stable while situation changes. However, it costs too much time for a CNN-based algorithm on a resource-constraint edge device to process an image since they are compute-heavy. This results in a low FPS when we implement them on the NVIDIA AGX Xavier [1] platform used in our closed-loop LKAS, which makes them unsuitable for our work.

On the other hand, traditional lane detection algorithms [26], [9], [6] are relatively light-weight and can achieve a high FPS on an edge device. Moreover, an approximated ISP pipeline can further improve $T_s$ since it skips some stages. But unlike CNN-based algorithms, they are sensitive to the changing between different situation. For example, different road layouts or types of lane markers can easily result in a misidentified line. The approximated ISP pipeline also introduces more sensing noise due to the trade-off between image quality and workload. As a result, a traditional lane detection algorithm is not robust enough to ensure vehicle safety for considered closed-loop LKAS. Figure 1.3 shows the accuracy and FPS comparison of traditional and CNN-based lane detection algorithms.

We find out that if we can configure the LKAS based on different situations, the lane detection algorithm can achieve higher accuracy. Thus, situations should be classified first, which can be realized by light-weight CNN-based classifiers. Moreover, approximated ISP pipelines can be utilized to reduce the time penalty brought by these classifiers. This approach skips stages in an ISP pipeline and reduce the runtime at the cost of image quality, and thus reduces the sampling period and increases QoC. This new strategy of LKAS design called hardware- and situation-aware sensing should achieve both hardware-efficient and situation-robustness,

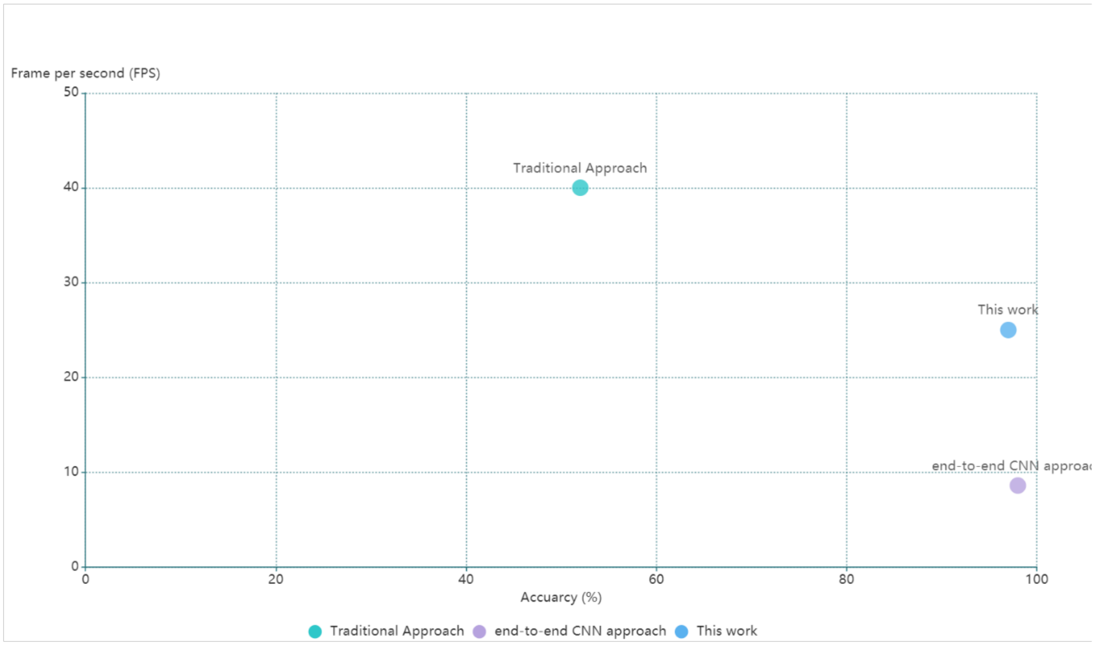

Figure 1.3: Accuracy and FPS comparison between different lane detection techniques

## 1.3 Problem Statement

The possible performance gains from hardware- and situation-aware sensing in an image-based control (IBC) system can have a significant impact on robustness and quality of control (QoC) of the system. The accuracy of lane detection algorithm affects the robustness. Hardware- and situation-aware sensing, which identifies different situations and dynamically reconfigure the IBC system, can increase lane detection accuracy, i.e. increase robustness. On the other hand, time of sensing task ($T_s$), which is the bottleneck for an IBC system on a resource-constraint device like NVIDIA AGX Xavier, is influenced by hardware- and situation-aware sensing as well. The time penalty brought by light-weight situation classifiers and gain brought by approximated ISP pipeline both affect the sampling period, which is equal to the time difference between two consecutive sensing tasks, i.e. affect QoC. The main focus of this project is to design hardware- and situation-aware perception for the IBC system and evaluate its impact on robustness and QoC on the system.

Therefore, the research question is: **Can we improve the robustness and quality of control of an image-based control system by designing hardware- and situation-aware perception?**

## 1.4 Contribution

We introduce light-weight CNN-based situation classifiers and configure the closed-loop LKAS dynamically based on the situational knowledge to make the hardware-efficient traditional approaches [5], [26] robust. Moreover, we also dynamically change the sampling period and sensor-to-actuation delay by using the approximated ISP pipelines and improve QoC. Thus, the new-designed closed-loop LKAS achieves both hardware-efficiency and situation-robustness with a high QoC.

The following paper [7] is published based on the results reported in this thesis. The paper is accepted for publication in Design Automation Test in Europe (DATE) 2021 special session on

Predictable Perception for Autonomous Systems.

- Sayandip De, Yingkai Huang, Sajid Mohamed, Dip Goswami, and Henk Corporaal, "Hardware- and situation-aware perception for robust closed-loop control systems," In Design, Automation and Test in Europe (DATE) special session on Predictable Perception for Autonomous Systems, 2021.

## 1.5 Report Structure

Chapter 1 is the introduction of the project. It contains motivation of this work, the problem statement and contribution. Chapter 2 discusses the related works on traditional sensing approach and end-to-end CNN-based approach. Chapter 3 illustrates the background of our work. Our purposed hardware-in-the-loop setup of LKAS, approximated ISP pipeline, lane detection algorithm, lateral control model and the considered hardware platform are all introduced in this chapter. Chapter 4 discusses the hardware- and situation-aware optimization we used in our work. It contains the full workflow, the definition of situations, hardware- and situation-aware Characterization, situation identification and dynamic runtime reconfiguration. Chapter 5 shows and analyzes the experimental result. Chapter 6 concludes the work and points out future research direction.

# Chapter 2

# Related Work

## 2.1 Traditional Sensing Approaches

These approaches use traditional methods to detect lanes, process images and perform approximate computing.

### 2.1.1 Traditional Lane Detection Algorithm

Many existing traditional algorithms are based on edge detection. [23] introduces a simple lane detection algorithm using Region of Interest (ROI), Canny edge detection and Hough Transform. [2] and [38] provide similar algorithms to [23]. [2] transforms the image from RGB to YCbCr and reduce the noise by an averaging filter. Differencing filter and Hough Transform are applied for edge detection. [38] constructs a lane geometrical model to gain lane geometrical features associated with the geometrical relationship between camera and road, which can improve the accuracy and reduce the computation in the lane detection process. [35] uses the bird's eye view (BEV) instead of a front-mounted camera. The image is transformed to grayscale first, and then the selected ROI is transformed to BEV via inverse perspective mapping(IPM) in order to determine lane detection region.

Although edge detection based algorithm is relatively mature now, it still has some limitations. When faced with different scenarios such as glare on the road, rainy and foggy weather and lane markers which are difficult to recognize, its performance will decrease.

### 2.1.2 Traditional Image Signal Processing Pipeline

The use of digital cameras is becoming significantly popular. To process the RAW images produced by the cameras, image signal processing (ISP) pipelines are widely researched. [31] introduces a digital color still camera (DSC) processing pipeline, which includes stages like demosaicing, denoising, color transforming and compression. [5] provides a similar traditional ISP pipeline and add functions such as gamut mapping and tone mapping to improve the quality of processed images. To enhance color saturation, [17] uses CIELAB color space in an ISP pipeline. This technology can increase the color saturation while keeping the hue and brightness of the pixels.

### 2.1.3 Approximate Computing in Image Processing

Approximated computing has been widely researched and implemented to achieve energy efficiency [12]. In [6], a scalable effort hardware design is proposed, which is based on Support Vector Machines (SVM). It can identify mechanisms at different levels of design abstraction such as circuit, algorithm and architecture and then make them scalable in the implementation. [14] provides a dynamic bitwidth adaptation in an Inverse Discrete Cosine Transform (IDCT) design. High-frequency components are processed in reduced-bitwidth adders since high-frequency coefficients have small magnitude values. This design can save 45% energy while losing 10dB peak signal-to-noise ratio (PSNR). These mentioned technologies focus on the processing algorithms and hardware implementation, which is hard to imply on an ISP pipeline.

[5] takes an different approach with [6] and [14]. Instead of using algorithm-level technologies, a reconfigurable image signal processing (ISP) pipeline is developed which can skip some complete stages. Although the quality of processed images is reduced, it has limited impact on the accuracy of vision tasks and can save 75% energy comparing to a complete traditional ISP pipeline. However, although the results of these approaches are promising, their evaluation is based on a particular dataset and does not consider a closed-loop system.

[8,9,26] considers image approximation for closed-loop image-based control systems. Here, the approximation scenarios are evaluated at design time, and one scenario is chosen to be implemented at runtime. For dynamically changing scenarios at runtime, the conclusion may not stand anymore. We will try to solve this problem by introducing a dynamically configured approximate ISP pipeline in this project.

## 2.2 End-to-end CNN-based Sensing Approaches

These approaches use CNN-based methods to detect lanes and process images.

### 2.2.1 End-to-end CNN-based Lane Detection Algorithms

In recent years, deep learning based algorithms are heavily researched due to the rapid development of the neural network. [20] presents a real-time lane-detection-and-tracking system which uses a classifier to separate the lane markers. In addition, it detects the left- and right-lane boundaries separately instead of using a fixed-width model. The system can therefore deal with challenging situations like worn lane markings and merging or splitting lanes. [11] introduces an end-to-end lane position estimation based on deep neural network, which can achieve 99% accuracy. [18] uses convolutional neural networks to enhance input images and extract ROI before performing RANSAC [19].

Another approach is made in [30], where a neural network end-to-end for lane detection referred to as LaneNet is developed. As Figure 2.1 shows, the input image is first encoded by a shared encoder, which is used for feature extraction. Then two decoders are purposed. One is for binary lane segmentation, which is used to an output binary segmentation map to identify lane pixels. The other one is for pixel embedding, which is trained, for instance, segmentation. Finally, these two featured branches are combined together and clustered to give the result of pixels of each line. LaneNet can achieve a 96.4% accuracy at 50fps measured on an NVIDIA 1080 TI.

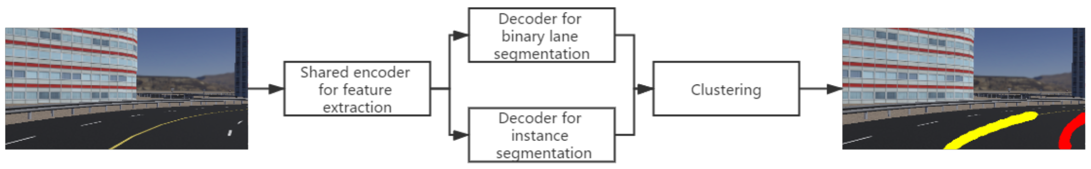
Shared encoder for feature extraction
Decoder for binary lane segmentation
Decoder for instance segmentation
Clustering

Figure 2.1: Framework of LaneNet

### 2.2.2 End-to-end CNN-based Image Signal Processing Pipeline

In recent years, deep learning also achieved success in image processing tasks. [32] presents a full end-to-end deep neural model of the camera image signal processing pipeline called DeepISP, which can perform different low-level corrections and higher-level global image restoration simultaneously. [15] applies deep learning technology to image processing approximation. For a new camera system, the ISP pipeline can be designed automatically and still provides high-quality images.

## 2.3 Shortcomings of Existing Approaches

Unfortunately, for our work, both traditional and CNN-based approaches have their own weakness. For traditional approaches, their robustness for different situations is not ensured because their configurations are static and cannot be aware of the situation changing. Although they are hardware-efficient, it is impossible to take the risk of crashing a vehicle. On the other hand, CNN-based approaches are more robust and adaptive to different situations. However, they are compute-heavy. For an edge device implemented in an autonomous vehicle, it will take too much time to perform one cycle and thus results in a low FPS. This will reduce the QoC of the system a lot and even crash the car as well.

To solve this problem, we consider a strategy to combine traditional and CNN-based approaches together. The traditional lane detection algorithm introduced in [26], [9] are used as backbone of the closed-loop system. To improve robustness, three different classifiers are trained to reconfigure the system based on the current situation the vehicle is running under. Moreover, knowledge of the approximated ISP pipeline introduced in [8, 9, 26] is also considered in the system to reduce the sampling period and sensor-to-actuation delay. This approach will be described in Chapter.

# Chapter 3

# Background

## 3.1 Hardware-in-the-Loop (HiL) Setup

The HiL setup of the considered LKAS in our work consists of two main parts, as Figure 3.1 shows. Webots, an open-source and multi-platform desktop application used to simulate robots, is used as the simulation server. It provides a complete development environment to model, program and simulate robots [24] and, more importantly, can be extended for automobile simulation. The simulation environment is built-in Webots as a .wbt world file. The camera equipped on the vehicle will send an image every sampling period to the client. The communication between server and client is completed through the TCP/IP protocol.

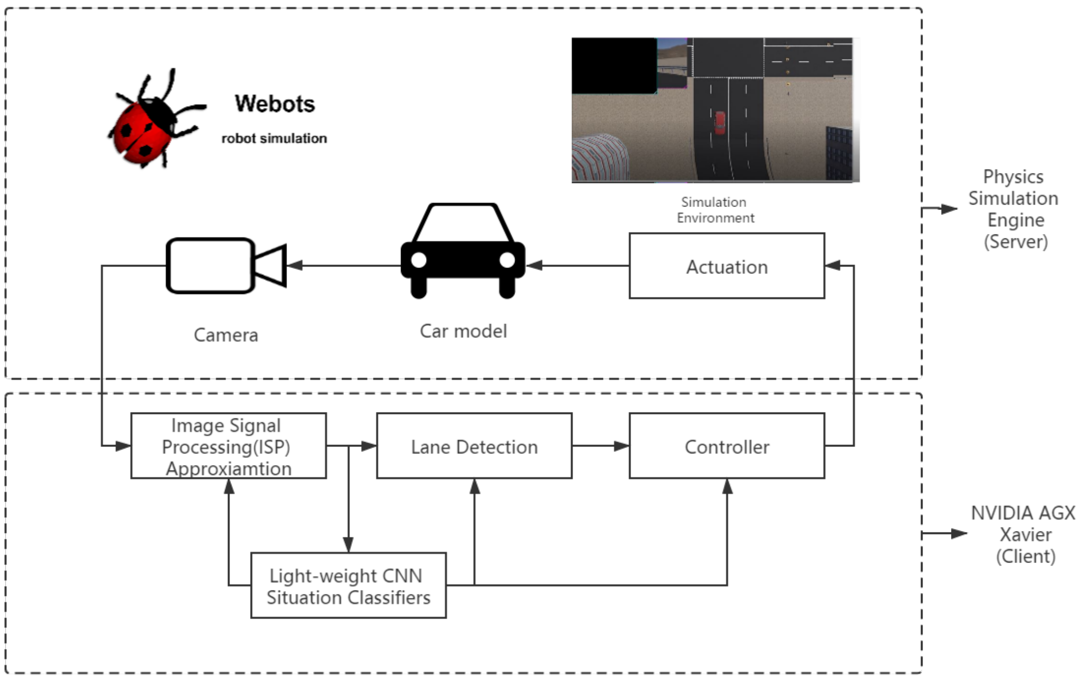
Webots
robot simulation
Simulation Environment
Physics Simulation Engine (Server)
Actuation
Car model
Camera
Image Signal Processing(ISP) Approxiamtion
Lane Detection
Controller
NVIDIA AGX Xavier (Client)
Light-weight CNN Situation Classifiers

Figure 3.1: HiL simulator setup

The client is executed in NVIDIA AGX Xavier platform and contains four components: ISP pipeline, perception, light-weight CNN classifiers and controller. The approximated ISP pipeline, which is based on Halide programming language, preprocesses the RAW image and pass it to the lane detection component. Then the perception part will process the image and calculate the lateral deviation. The controller will calculate the required steering angle based on the lateral deviation and send it back to Webots to keep the vehicle staying in the middle of the lane. Several light-weight CNN classifiers, which will be explained in Chapter 4.4, are also introduced to dynamically tune the configuration of ISP pipeline, perception and controller based on the environmental situations and hardware settings.

## 3.2 Approximated Image Signal Processing Pipeline

### 3.2.1 Image Signal Processing Pipeline

ISP technology largely determines the imaging quality since cameras currently have physical imperfections and need to adapt to the environment because of the variety of light conditions for shooting. By performing subsequent processing on the RAW signal output by the front-end image sensor, ISP can generate compressed images which satisfy human vision.

A traditional ISP pipeline consists of a series of signal processing stages [5]:

1. Demosaicing: Bayer Color Filter Array (CFA) is commonly used on cameras. As a result, each pixel in the RAW output image only contains one color: red, green or blue. Demosaicing can obtain the composition value of red, green and blue of each pixel through interpolation and generate an RGB image.

2. Denoising: Each sensor contains an analog part, so noise in the signal is difficult to avoid. In addition, when the light condition is low, the entire system needs to amplify the signal, which means the noise is also amplified. By averaging the neighboring pixels that resemble each other, denoising can improve the signal-to-noise ratio.

3. Color transform: This stage mainly contains two parts, color mapping and white balancing. The former attempts to make the intensity of RGB matched while the latter adjusts the color temperature of the entire picture depending on light condition.

4. Gamut mapping: The pixel values are corrected to be within a display's acceptable color range in this stage.

5. Tone mapping: The image collected by the camera sometimes can not display enough details in the darker or brighter parts at a certain exposure. Tone mapping raises pixel values in particularly dark areas and reduces them in particularly bright areas in order to reproduce details in these places.

6. Compression: Finally, the image will be compressed to reduce the data storage and communication cost when it is transferred to another processing unit.

Figure 3.2 shows the workflow of a traditional ISP pipeline.

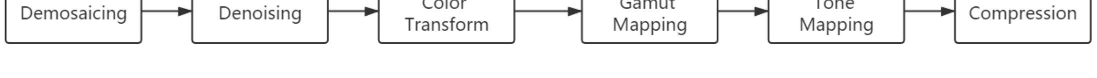

Figure 3.2: Traditional ISP pipeline

### 3.2.2 Approximated ISP pipeline

Although the traditional ISP pipeline has a great effect on the image quality, it requires more calculation time and energy than an approximate ISP pipeline. Some stages may be unnecessary for some computer vision applications so they can be skipped to improve the performance of ISP pipeline. In this project, a tool called Configurable & Reversible Imaging Pipeline (CRIP) developed in [5] is implemented. CRIP can simulate an imaging pipeline in "forward" operation and invert the function in "reverse" mode. By CRIP, different configurations of stages can be simulated and evaluated in the application. Nine versions of pipelines are defined in this project

| Version | Stages | Description |
|---|---|---|
| S0 | DM, DN, CT, GM, TM | All stages included |
| S1 | DM. DN, CT, TM | Skip gamut mapping |
| S2 | DM, DN, CT, GM | Skip tone mapping |
| S3 | DM, DN, GM, TM | Skip color transform |
| S4 | DM, CT | Only keep color transform |
| S5 | DM, GM | Only keep gamut mapping |
| S6 | DM, TM | Only keep tone mapping |
| S7 | DM, DN | Only keep denoising |
| S8 | DM, CT, GM, TM | Skip denoising |

Table 3.1: Approximated ISP pipeline configuration

as Table 3.1 shows. S0 is the accurate version which will take the most time, the other eight versions all skip some stages according to their configuration. Skipping the demosaic stage will cause a crash because perception algorithm does not consider the Bayer pattern while skipping compression will decrease energy efficiency, so demosaicing and compression are performed in all these versions. The output images of these nine versions of pipelines are shown in Figure 3.3

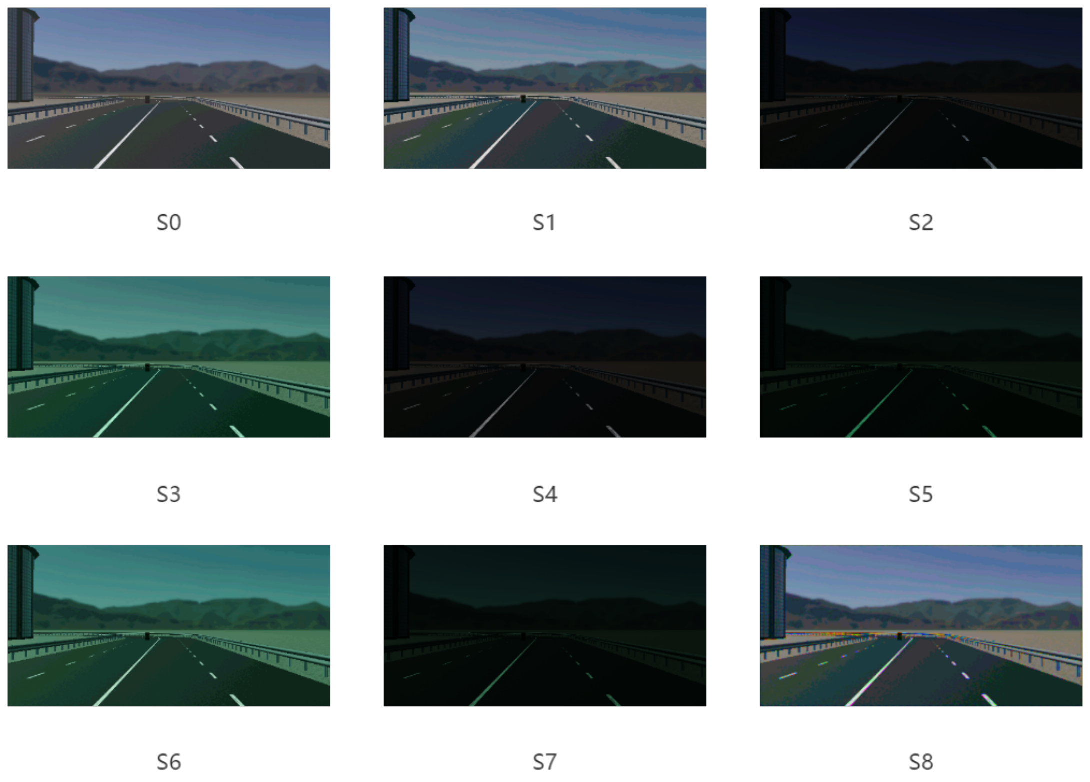

Figure 3.3: Output of Nine Approximated ISP Pipelines

## 3.3 Lane Detection Algorithm

The approximated ISP pipeline sends the preprocessed image to this process unit. Lane detection algorithm mainly has three stages: perspective transform, feature extraction and inference as Figure 3.4 shows.

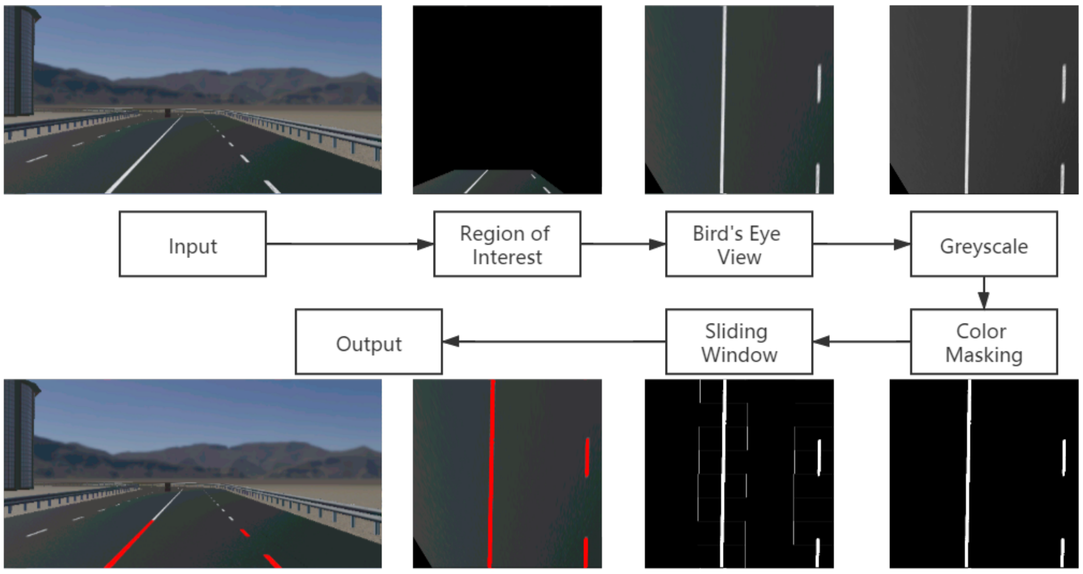

Figure 3.4: Workflow of Lane Detection

### 3.3.1 Perspective Transform

The image provided by the approximated ISP pipeline contains a lot of unnecessary information such as road lines of other lanes, buildings and debris off the road. So four region of interest (ROI) points are selected to obtain the useful part of the image. For different cameras, these four points should be different. Besides, when the vehicle is performing a left turn or right turn, its ROI needs to be changed as well because the position of two road lines on the images slides to left of right.

Then the image is transformed to get a bird's eye view (BEV) of the look-ahead lane. Table 3.2 lists the four BEV points.The four ROI points are mapped to the four defined BEV points as Figure 3.5 shows. Finally, the BEV image is converted to grayscale in this stage.

| | |
|---|---|
| x1d | (120,512) |
| x2d | (392,512) |
| x3d | (120,0) |
| x4d | (392,0) |

Table 3.2: BEV points

### 3.3.2 Feature Extraction

To make the white road lane markings clearly differentiable from the road, color masking is then performed. A threshold is set in this stage. If a pixel is brighter than this threshold, it is set to the highest brightness, and otherwise, it becomes darkest. Sliding window based lane detection follows behind, and tracks left and right road lines by sliding windows from bottom to top of the image.

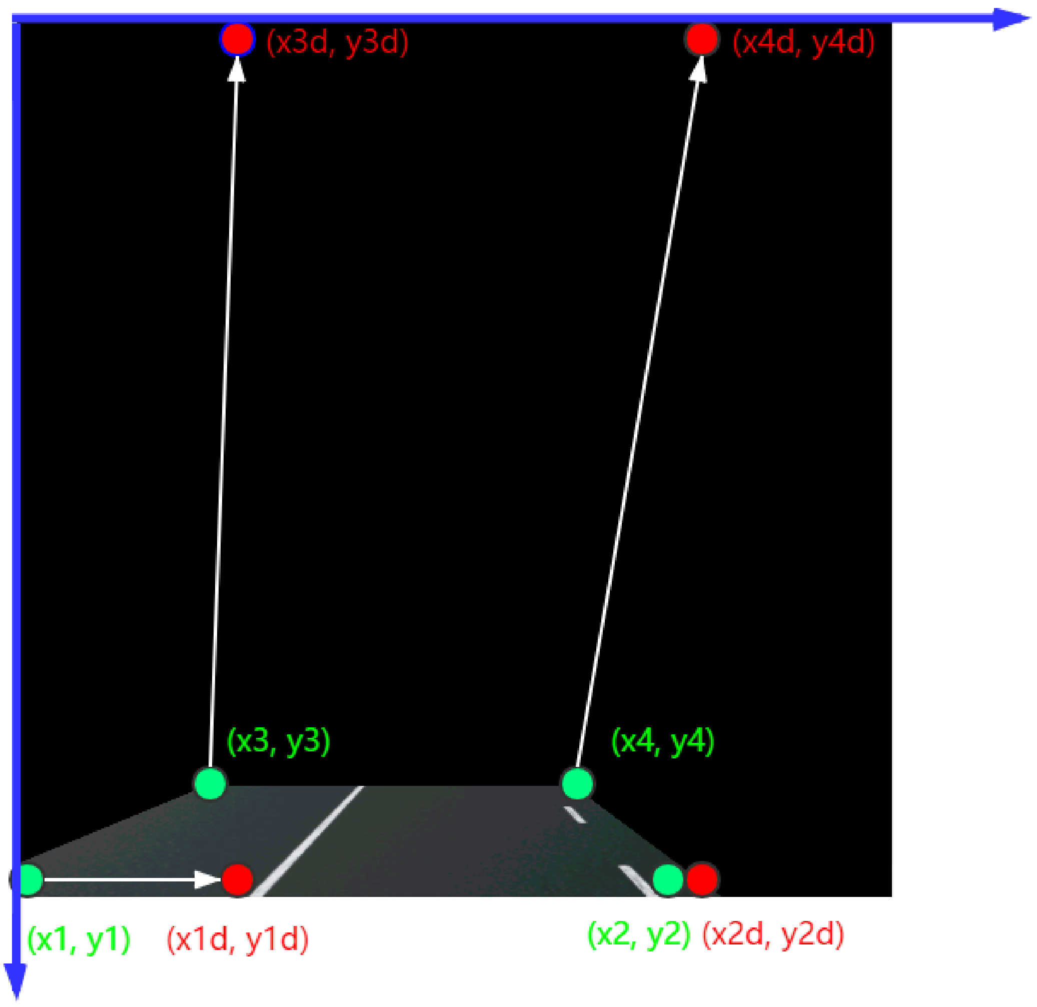

Figure 3.5: Perspective Transform

### 3.3.3 Inference

The last step of lane detection is lateral deviation calculation. The tracked left and right road lines are fit to a second degree polynomial. Then the center of the lane is identified using these polynomials based on the look-ahead distance, which is 5.5 meters in this project. After that, the reference of current vehicle position can be calculated according to :

$$ref = x_{1d} + \frac{(x_{2d} - x_{1d})(256 - x_1)}{x_2 - x_1} \tag{3.1}$$

where $x_1$ and $x_2$ are abscissa of ROI points 1 and 2 shown in Figure 3.5, $x_{1d}$ and $x_{2d}$ are abscissa of BEV points 1 and 2 shown in Figure 3.5.

Finally, the lateral deviation is calculated according to:

$$y_L = (ref - lanecenter) * \frac{l_m}{l_p} \tag{3.2}$$

where $l_m$ is lane width in meter and $l_p$ is lane width in pixel.

## 3.4 Lateral Control

The control task receives the lateral deviation ($y_L$) at look-ahead distance from lane detection algorithm and calculates the steering angle $\delta_f$. The lateral controller is defined as a linear time-invariant (LTI) system:

$$\dot{x} = Ax(t) + Bu(t),\ y(t) = Cx(t) \tag{3.3}$$

where $x(t)$, $u(t)$ and $y(t)$ represent the state, input and output of the system. The lateral controller used in this work is adapted from [25, 27–29, 37].

The state vector $x(t) = \begin{bmatrix} v_y & \psi & y_L & \epsilon_L & K_L \end{bmatrix}^T$ where $v_y$ is the lateral velocity, $\psi$ is the yaw rate, $y_L$ is the lateral deviation, $\epsilon_L$ is the angle between the tangent of the road and the vehicle orientation, $K_L$ is the curvature of the road at look-ahead distance.

The input $u(t)$ is the front wheel steering angle $\delta_f$ and the output $y(t)$ is $y_L$. A, B, C represent the state, input and output matrices of the system, and these matrices are:

$$A = \begin{bmatrix} -\frac{c_f+c_r}{mv_x} & \frac{-mv_x^2+c_rl_r-c_fl_f}{mv_x} & 0 & 0 & 0 \\ \frac{c_rl_r-c_fl_f}{I_\psi v_x} & \frac{c_rl_r^2-c_fl_f^2}{I_\psi v_x} & 0 & 0 & 0 \\ -1 & -L & 0 & v_x & 0 \\ 0 & -1 & 0 & 0 & v_x \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} \frac{c_f}{m} \\ \frac{l_fc_f}{I_\psi} \\ 0 \\ 0 \\ 0 \end{bmatrix}, C = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.4}$$

where $c_f$ and $c_r$ (= 2×60000N/rad) are the cornering stiffness of the front and rear tires,
$m$ (=2000kg) is the mass of the vehicle,
$v_x$ is the longitudinal velocity,
$l_f$ and $l_r$ (= 1:6975 and 1:2975 m respectively) are the front and rear axle-center of gravity (CoG) distance,
$I_\psi$ (= 6337.74kg·m$^2$) is the total inertia of vehicle around CoG,
$L$ is the look-ahead distance of the camera.

The lateral controller needs to be discretized using the Zero-Order Hold (ZOH) method since it is implemented on a digital platform. The lateral controller also needs to take the worst sensor-to-actuator delay($D_c$) into account. The sampling period($h$) should always be larger than $D_c$. The sensor-to-actuator delay is obtained by first profiling the individual tasks and then summation of their profile times. Timing analysis [3] is done to obtain the delay and period for different mapping options. The system is then modeled considering the delay and period as:

$$x[k+1] = A_dx[k] + B_0(D_c)u[k] + B_1(D_c)u[k-1] \tag{3.5}$$

where $A_d = e^{Ah}$, $B_0(h) = \int_0^{h-D_c} e^{A_cs}ds \cdot B_c$, $B_1(h) = \int_{h-D_c}^{h} e^{A_cs}ds \cdot B_c$.

An augmented state is introduced such that $z[k] = [x(k)u[k-1]]$. Assuming $z[0] = [x(0)0]$, a higher-order augmented system is shown as:

$$z[k+1] = A_{aug}(h)z[k] + B_{aug}(h)u[k] \tag{3.6}$$

where $A_{aug} = \begin{bmatrix} A & B_1(h) \\ 0 & 0 \end{bmatrix}$, and $B_{aug} = \begin{bmatrix} B_0(h) \\ I \end{bmatrix}$

The objective of the lateral controller is getting an input $u[k]$ which can make output y[k] reach a constant reference as k tends to infinity. In the LKAS, the reference is *zero*.

Due to dynamic reconfiguration at runtime, the controllers are switching based on different situations. Each situation $i$ has a controller configuration defined by $h_i,\ D_{c_i}$. The stability of the switched control system is shown by the existence of a common quadratic Lyapunov function as explained in [25, 27].

## 3.5 Considered Platform

### 3.5.1 Platform Description

NVIDIA AGX Xavier platform, an edge device with a maximum power budget of 30W [1], is used to execute LKAS as a client. The used parts in this project. an 8-core NVIDIA Carmel ARMv8.2 CPU, an integrated 512-core NVIDIA Volta GPU and 16GB of LPDDR4x off-chip DRAM memory, is shown in Figure 3.6.

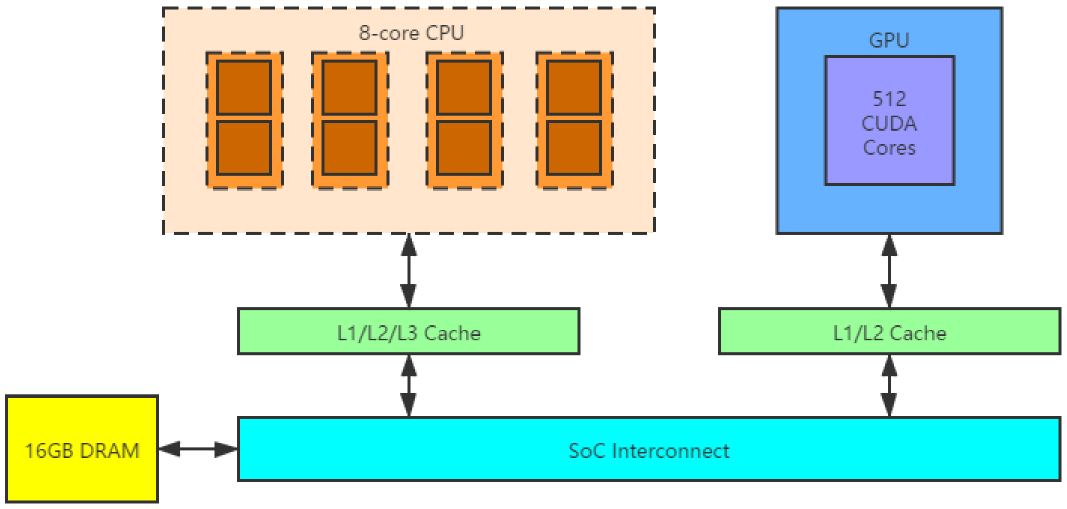

Figure 3.6: Framework of NVIDIA AGX Xavier Platform

### 3.5.2 Task Mapping

Table 3.3 shows the CPU-GPU task mapping on the client. Since the ISP pipeline is computing-heavy, all the ISP tasks are mapped to the GPU except input data loading. Besides, ROI selection, perspective transform and image thresholding of perception are also mapped to the GPU to help reduce the running time. The actuation part is mapped to the CPU because it is relatively light in computation.

| | CPU | GPU |
|---|---|---|
| ISP | Loading input data | Demosaicing |
| | | Denoising |
| | | Color Mapping |
| | | Gamut Mapping |
| | | Tone Mapping |
| Perception | Sliding window based tracking | ROI selection & Perspective transform |
| | yL calculation | Image thresholding |
| Actuation | Steering angle computing | |

Table 3.3: Task Mapping on the Client

# Chapter 4

# Hardware- and Situation-aware Sensing

## 4.1 Step-wise Overview of Hardware- and Situation-aware Method

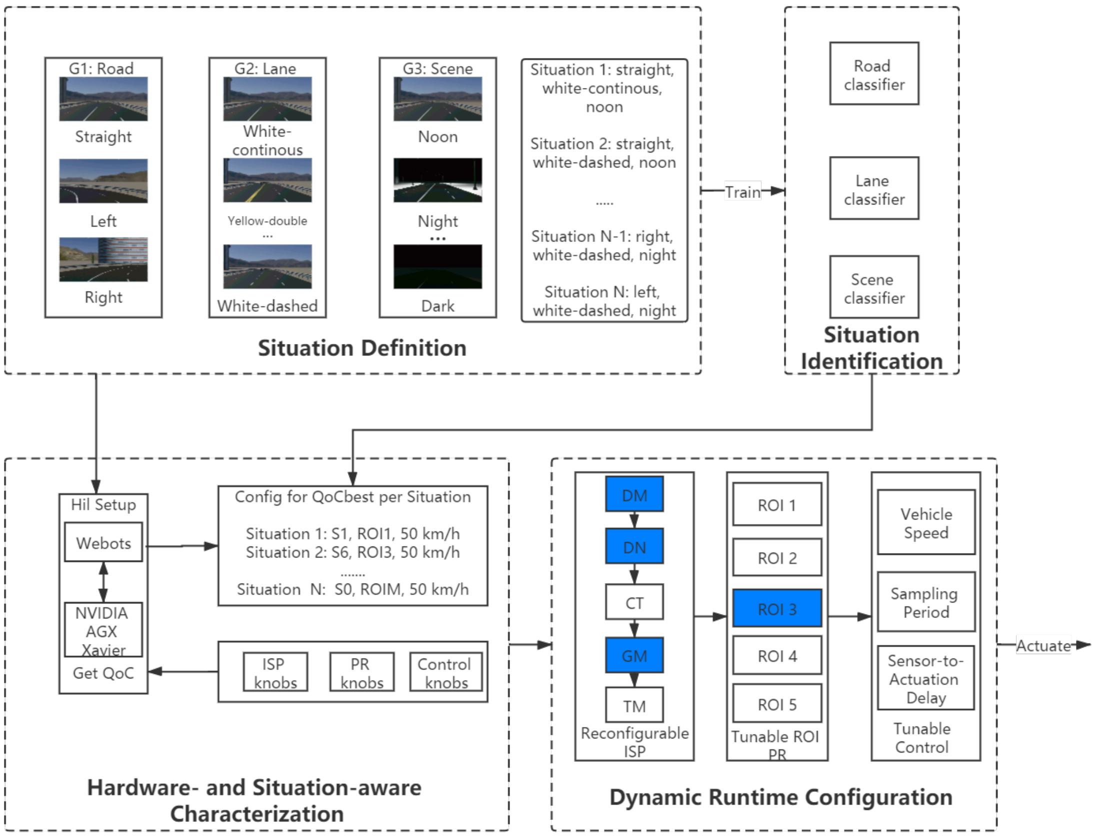
G1: Road
Straight
Left
Right
G2: Lane
White-continous
Yellow-double
White-dashed
G3: Scene
Noon
Night
Dark
Situation 1: straight, white-continous, noon
Situation 2: straight, white-dashed, noon
Situation N-1: right, white-dashed, night
Situation N: left, white-dashed, night
Situation Definition
Train
Road classifier
Lane classifier
Scene classifier
Situation Identification
Hil Setup
Webots
NVIDIA AGX Xavier
Get QoC
Config for QoCbest per Situation
Situation 1: S1, ROI1, 50 km/h
Situation 2: S6, ROI3, 50 km/h
Situation N: S0, ROIM, 50 km/h
ISP knobs
PR knobs
Control knobs
Hardware- and Situation-aware Characterization
DM
DN
CT
GM
TM
Reconfigurable ISP
ROI 1
ROI 2
ROI 3
ROI 4
ROI 5
Tunable ROI PR
Vehicle Speed
Sampling Period
Sensor-to-Actuation Delay
Tunable Control
Actuate
Dynamic Runtime Configuration

Figure 4.1: Step-wise Overview of Hardware- and Situation-aware Method

Figure 4.1 shows the step-wise overview of our proposed hardware- and situation-aware method. First, we will define different situations based on three main features. Then hardware- and situation-aware characterization will be used to find out the specific configuration of LKAS which can achieve the best robustness and QoC. After this, three light-weight CNN-based situation classifiers are trained and integrated into the LKAS system. Finally, we will design a dynamic runtime reconfiguration workflow for the considered LKAS.

## 4.2 Situation Definition

In this project, three different features are considered to have the most impact on robustness and QoC: road layouts, type of lane markers and type of scene/weather.

### 4.2.1 Road Layouts

In real scenarios, a vehicle does not always drive in one direction and needs to turn left and right in different curves. However, it is hard to configure a static ROI for all types of turns. For example, the ROI fit for the straight road cannot contain both two road lines in a right turn and will cause a vehicle crash, which is shown in Figure 4.2. Therefore, a dynamic configured ROI is necessary for a realistic environment. Three different road layouts are defined: straight, left turn and right turn, as shown in Figure 4.3.

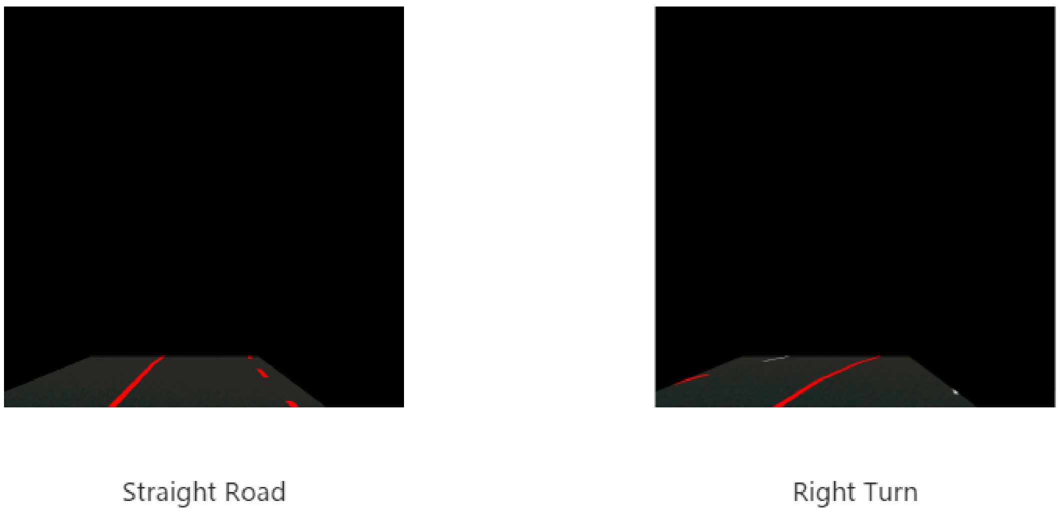

Figure 4.2: Static ROI's Performance on Straight Road and Right Turn

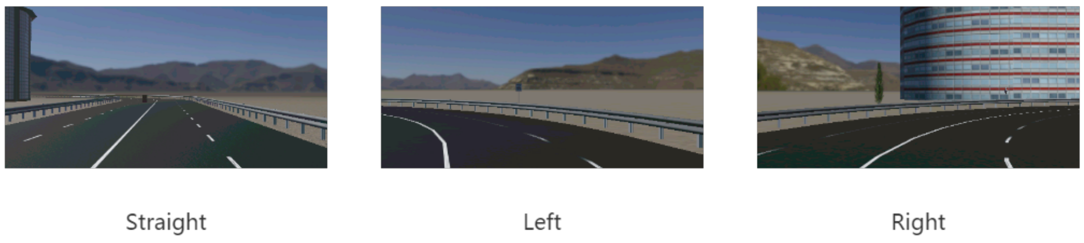

Figure 4.3: Different Road Layouts

### 4.2.2 Type of Lane Markers

Different forms of lane markers are widely used in real life, such as continuous line, double line and dashed line. Some of these lane markers have an effect on the robustness of LKAS. After the perspective transform, dashed lines are harder for the sliding window algorithm to recognize than continuous and double lines, especially in the curve. Moreover, the color of lines also affects the QoC since for some approximation ISP pipelines, the quality of their output image reduces. Thus, one type of lane markers should be a combination of form and color. In this project, four kinds of lane markers are considered: white-continuous line, white-dashed line, yellow-continuous line, yellow-double line, as shown in Figure 4.4.

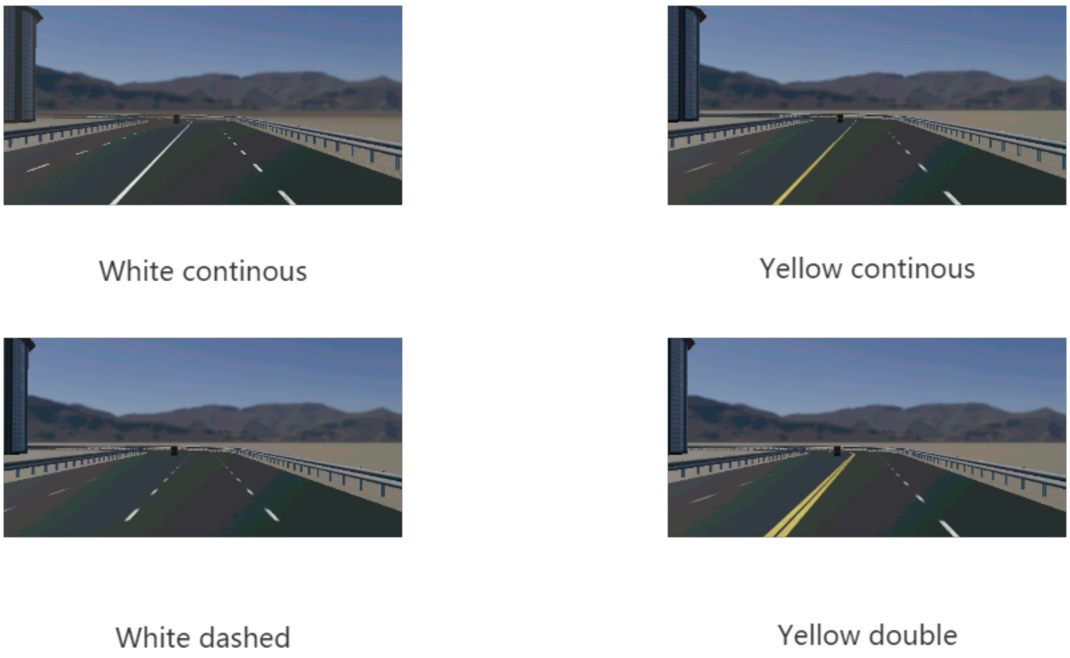

Figure 4.4: Different Types of Lane Markers

### 4.2.3 Type of Scene

In normal light conditions, the average grey value of the image is relatively uniform. However, in real life, luminosity will not always remain ideal and vary with the weather and time, which will make lane detection more difficult. To give LKAS information of the current situation, which can help ISP pipeline to provide more accurate images facing with these scenarios, five different scenes are considered: noon, dusk, dawn, night (with streetlight) and dark (without streetlight), as shown in Figure 4.5.

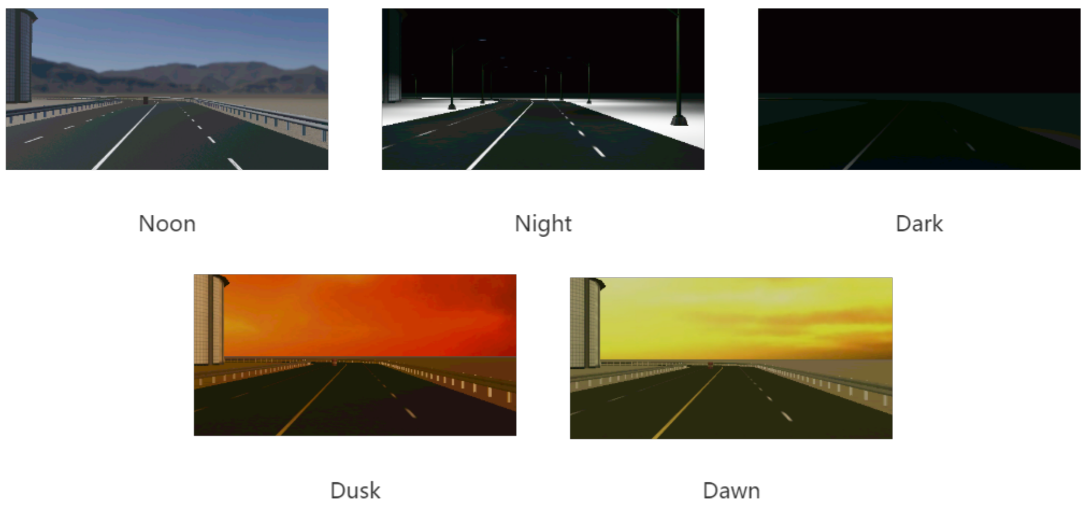

Figure 4.5: Different Types of Scenes

## 4.3 Hardware- and Situation-aware Characterization

### 4.3.1 Considered Configurable Knobs in the System

Hardware- and situation-aware characterization will identify the set of LKAS parameters which perform best under a specific situation at design time. To achieve this goal, several system parameters which are sensitive to the situation are described in Table. The nine ISP pipeline versions, which are already discussed in Chapter 4, will have a big effect on the sensor-to-actor delay and sampling delay. Besides, five different ROIs, whose pixels are reported for 512X256 resolution frames, are defined to improve the robustness of LKAS. Moreover, different vehicle speeds are also considered since the high speed will bring more danger in the curve. Table 4.1 shows all the considered knobs in the system.

| Knobs | Detailed list | Runtime |
|---|---|---|
| ISP knobs | S0:(DM, DN, CT, GM, TM) | 21.5ms |
| | S1:(DM. DN, CT, TM) | 3.3ms |
| | S2:(DM, DN, CT, GM) | 3.2ms |
| | S3:(DM, DN, GM, TM) | 20.9ms |
| | S4:(DM, CT) | 3.2ms |
| | S5:(DM, GM) | 3.1ms |
| | S6:(DM, TM) | 3.2ms |
| | s7:(DM, DN) | 3.1ms |
| | S8:(DM, CT, GM, TM) | 18.9ms |
| PR knobs | ROI 1: (60, 0) (300, 0) (160, 65) (280, 65)<br>ROI 2: (208, 0) (469, 0) (308, 72) (439, 72)<br>ROI 3: (188, 0) (469, 0) (298, 72) (429, 72)<br>ROI 4: (69, 0) (333, 0) (117, 72) (221, 72)<br>ROI 5: (49, 0) (312, 0) (109, 72) (222, 72) | 3.0ms |
| Control knobs | vehicle speed: 30km/h, 50km/h<br>sampling period(h), sensor-to-actuation delay($\tau$) | 2.5us |

Table 4.1: Considered configurable knobs in the system

| Sit.No. | Road layout | Left lane marker type | Scene | Speed | ISP knobs | PR knobs | $T_c$ knobs [h,$\tau$](ms) |
|---|---|---|---|---|---|---|---|
| 1 | straight | white-continuous | noon | 50km/h | S1 | ROI1 | [25,23.1] |
| 2 | straight | white-dashed | noon | 50km/h | S5 | ROI1 | [25,22.4] |
| 3 | straight | yellow-continuous | noon | 50km/h | S2 | ROI1 | [25,22.5] |
| 4 | straight | yellow-double | noon | 50km/h | S4 | ROI1 | [25,22.5] |
| 5 | straight | white-continuous | night | 50km/h | S4 | ROI1 | [25,22.5] |
| 6 | straight | yellow-continuous | night | 50km/h | S6 | ROI1 | [25,23.0] |
| 7 | straight | white-continuous | dark | 50km/h | S6 | ROI1 | [25,23.0] |
| 8 | right | white-continuous | noon | 30km/h | S4 | ROI2 | [25,22.5] |
| 9 | right | yellow-continuous | noon | 30km/h | S1 | ROI2 | [25,23.1] |
| 10 | right | yellow-double | noon | 30km/h | S1 | ROI2 | [25,23.1] |
| 11 | right | white-continuous | night | 30km/h | S6 | ROI2 | [25,23.0] |
| 12 | right | yellow-continuous | night | 30km/h | S1 | ROI2 | [25,23.1] |
| 13 | right | white-dashed | noon | 30km/h | S1 | ROI3 | [25,23.1] |
| 14 | right | white-dashed | night | 30km/h | S6 | ROI3 | [25,23.0] |
| 15 | left | white-continuous | noon | 30km/h | S1 | ROI4 | [25,23.1] |
| 16 | left | yellow-continuous | noon | 30km/h | S6 | ROI4 | [25,23.0] |
| 17 | left | yellow-double | noon | 30km/h | S6 | ROI4 | [25,23.0] |
| 18 | left | white-continuous | night | 30km/h | S1 | ROI4 | [25,23.1] |
| 19 | left | yellow-continuous | night | 30km/h | S6 | ROI4 | [25,23.0] |
| 20 | left | white-dashed | noon | 30km/h | S3 | ROI5 | [45,40.7] |
| 21 | left | white-dashed | night | 30km/h | S3 | ROI5 | [45,40.7] |

Table 4.2: Pre-characterized situation-specific knob tunings

### 4.3.2 Pre-characterized Situation-specific Knob Tunings

Table 4.2 shows the 21 different situations considered in this project. For straight roads, the vehicle speed is 50 km/h while in the curves it is reduced to 30 km/h. This is because 50 km/h is too dangerous for a car to stay in a curve with a small radius. Such a high speed needs a smaller sampling period and delay, which is hard to achieve in this work.

To determine which knob tuning is can achieve the best QoC for each situation, MAE is introduced in this project. MAE can be calculated as the average distance from the position of the vehicle to the middle of lane and the knob tuning whose MAE is the smallest is considered as the optimal one.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y[k]| \tag{4.1}$$

where $n$ is the no. of samples and $y[k]$ is the value $y_L$ at the $k^{th}$ sample and ideally $y_L$ should be *zero*.

Figure 4.6, 4.7 and 4.8 show the MAEs of different ISP knob tunings for straight, right turn and left turn respectively. For straight roads and right turns, the ISP knobs with smaller sampling periods perform better than those with bigger periods. However, when the left lane marker is dashed in a left curve, S3 has the best QoC because it is difficult to track dashed lines in the presence of approximation. The ROI is also changed for situations with the dashed left line in a curve, which results in a worse QoC comparing to other situations. On the other hand, for curves, several ISP pipeline versions will lead the vehicle to crash since they skip some specific stages which are important for lane detection in a curve. For example, under situations with a left curve, ISP pipeline version S2, S4, S5 and S7 will be likely to lead the car to crash. All these four versions skip tone mapping, and thus, we can come to such a conclusion that tone mapping is important to a left turn.

The final knob tunings are shown in Table 4.2. For situation 13, 14, 20 and 21, their ROI is different from the situations with the same road layouts. This is due to the two dashed lines in a curve which will lead the car to crash quickly. Thus a different ROI which is more stable in curves is used, although it will bring a higher MAE.

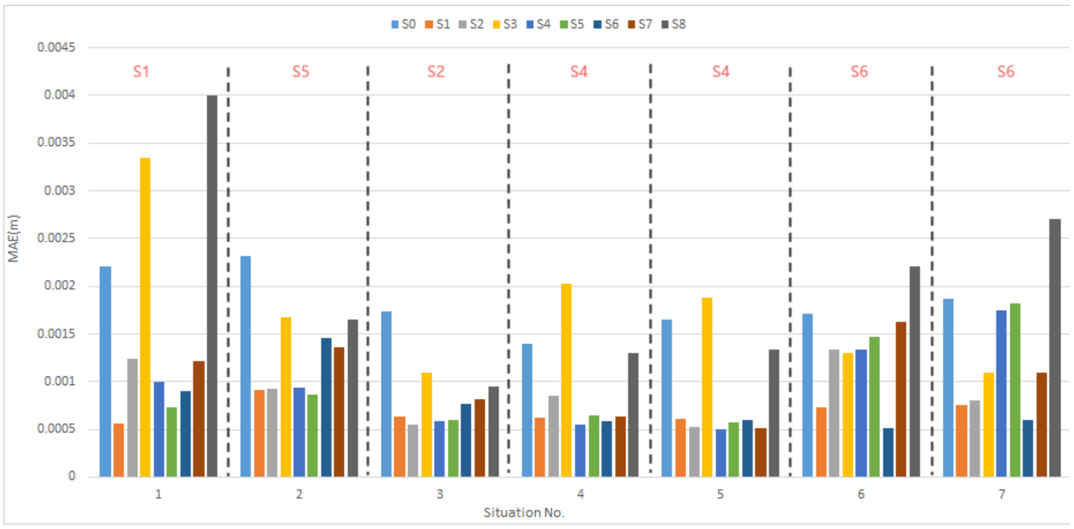

Figure 4.6: MAE of Different ISP Knob Tunings for Straight Road

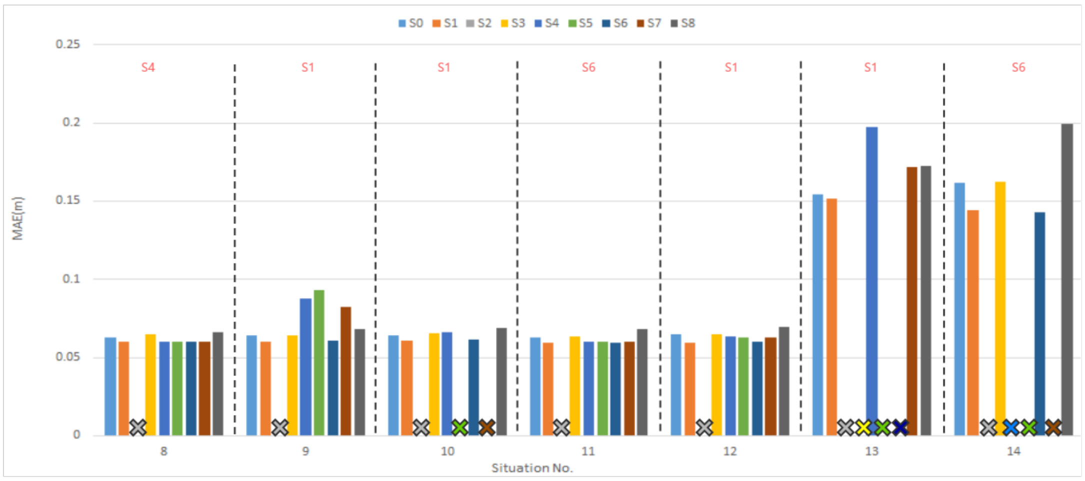

Figure 4.7: MAE of Different ISP Knob Tunings for Right Turn

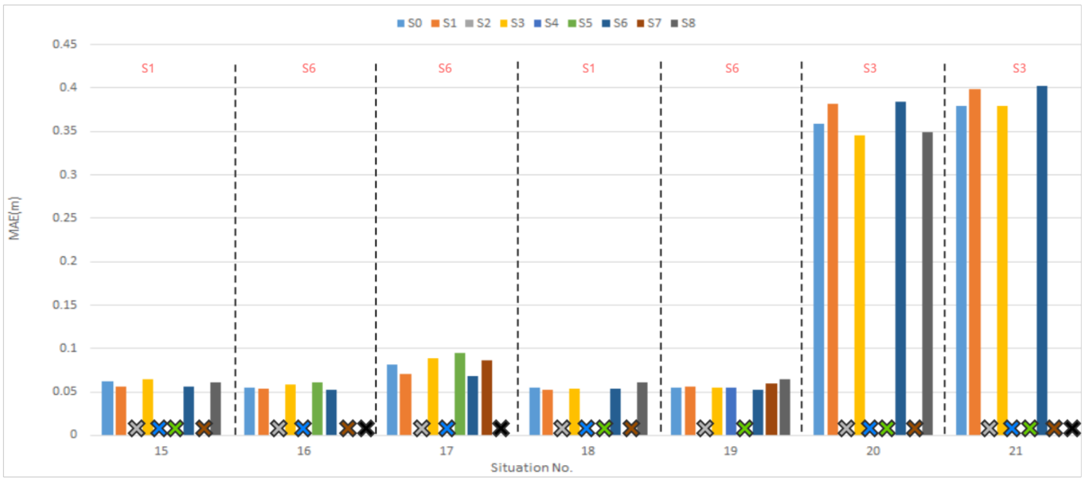

Figure 4.8: MAE of Different ISP Knob Tunings for Left Turn

## 4.4 Situation Identification

The basic idea of situation classifiers is using transfer learning for computer vision to identify different situations based on images caught by camera sensors on the vehicle. In practice, it is difficult to get a sufficient dataset to train a new Convolutional Network(ConvNet).As a result, transfer learning, which uses a ConvNet pretrained on a very large dataset like ImageNet [10] either as an initialization or a fixed feature extractor, is commonly utilized. In this project, the strategy of transfer learning is loading a pretrained model and resetting final fully connected layer. Resnet18 [13] is selected as the pretrained model because of its good performance and high accuracy. The structure of resnet18 is shown as Figure 4.9

To improve the accuracy of classifiers, the images need to be preprocessed before being used as training data. The size of the original image provided by the camera in Webots is 512*256. However, the road is only shown in the bottom part of the image. Therefore the image is cropped to a size of 512*100 and then resized to 224*224 due to the requirement of resnet18. Then the pretrained resnet18 model is loaded, and the final fully connected layer is reset. After 50 epochs of training, the model with the best accuracy is saved. The workflow of transfer learning is shown in Figure 4.10.

Three different classifiers are trained to output information about the current situation. Table 4.3 shows the brief detail of each classifier. The training processing is based on Pytorch 1.4.0 with CUDA 10.0. The running time is profiled on the NVIDIA AGX Xavier platform. Since the dataset is all based on Webots, the accuracy can reach 99.90% at least.

| Classifiers | Function | Dataset | Output classes | Accuracy | Runtime |
|---|---|---|---|---|---|
| Road classifier | recognizes road layout | 5866 images (train:5353, val:513) | straight, left turn, right turn | 99.92% | 5.5ms |
| Lane classifier | recognizes lane type & color | 4781 images (train:3939, val:842) | white-dashed, white-continous, yellow-double, yellow-continous | 99.97% | 5.5ms |
| Scene classifier | recognizes operating scene/weather | 4703 images (train:3892, val:811) | day, night, dark, dawn, dusk | 99.90% | 5.5ms |

Table 4.3: Brief details of situation classifiers

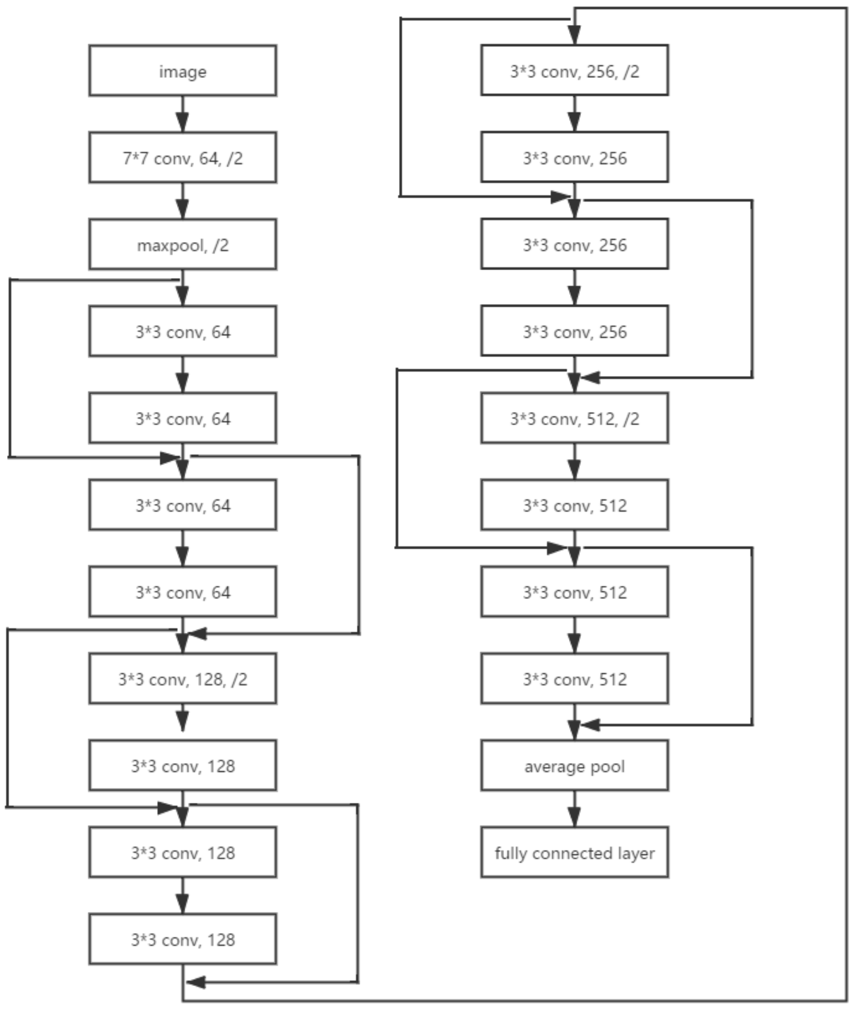

Figure 4.9: Structure of Resnet 18

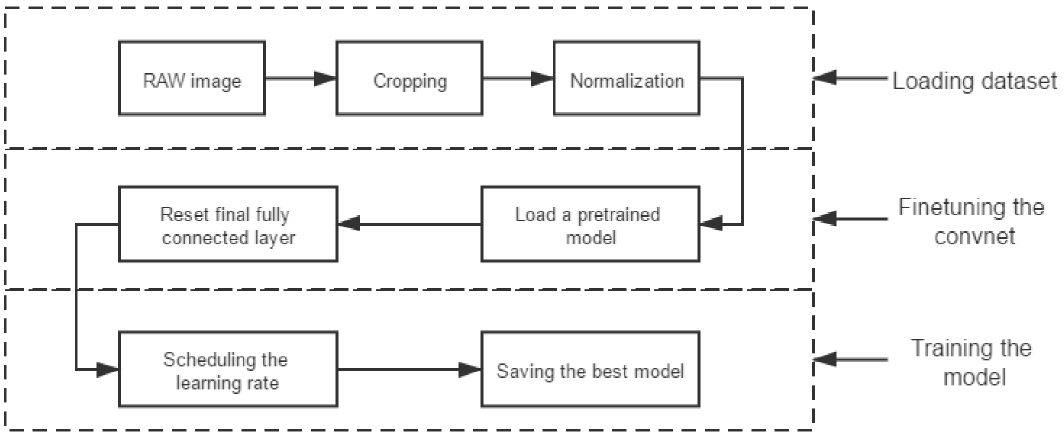

Figure 4.10: Workflow of Transfer Learning

## 4.5 Dynamic Runtime Reconfiguration

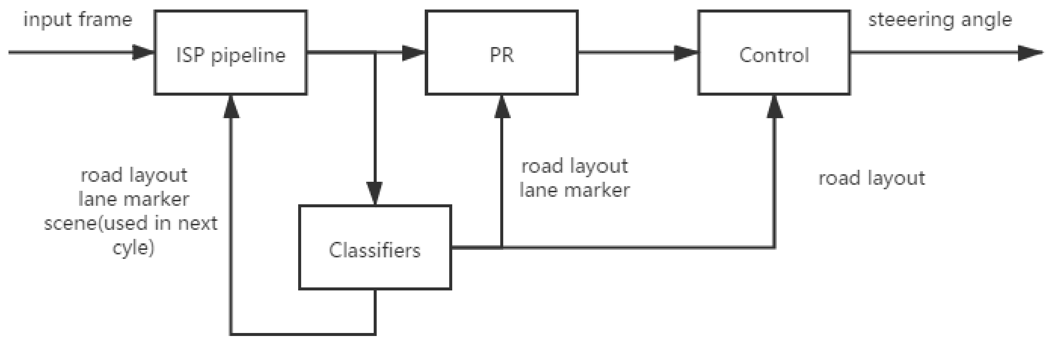

Figure 4.11: Workflow of Dynamic Runtime Reconfiguration

The three light-weight CNN situation classifiers introduced in Chapter 4.4 will be used to configure the knobs in LKAS dynamically. The workflow is shown in Figure 4.11. The ISP pipeline will first process the input frame and then pass it to the three classifiers. The output of classifiers will configure the PR and control knobs in the same cycle. However, the ISP knob will be configured in the next cycle. Since in real life, the situation does not change per frame, such an one-cycle delay will not result in a vehicle crash. For example, while operating at 40 FPS and vehicle speed of 50 km/h, the vehicle progresses only 35 cms per frame, which is well below the look-ahead distance (LL = 5.5 m) considered for designing the controller. This will also be certified in Chapter 5. All three classifiers decide the ISP pipeline version suitable for the current situation, and for perception, ROI is decided by road classifier and lane classifier. Only road classifier will directly have an impact on the control part since it will decide the vehicle speed alone.

# Chapter 5

# Experimental Results

This chapter will discuss the evaluation of hardware- and situation-aware method introduced in Chapter 4.

## 5.1 Experimental Settings

The Webots version used in the project is R2020a reversion 1. For evaluation, the camera frame rate is considered as 200 FPS in Webots. Lane width is always set to 3.25m according to standard road safety guidelines. Although the type of left lane marker may change with different situations, the right lane marker will always remain white-dashed, which is more common in real life. The time step of Webots simulation is set to 5 ms. To ensure that the correct frames are captured, and the correct steering angles are actuated, the sensor-to-actuation delay and sampling period are ceiled to the nearest factor of 5 ms.

To evaluate the closed-loop QoC of LKAS, MAE, which is introduced in sector 4.3.2, is again used as the evaluation metric.

## 5.2 Static Situation-specific Results and Analysis

### 5.2.1 Considered cases

Four cases, whose details are shown in Table 5.1, are considered to motivate the necessity of hardware- and situation-aware sensing for improving the robustness of LKAS. Case 1, which is used as the baseline, has the smallest sampling period since there are no classifiers in the LKAS. In case 2, a road classifier is added to tune the ROI and vehicle speed dynamically. When the classifier recognizes that the current road layout is a curve, the vehicle will slow down to 30 km/h; otherwise, it will keep 50 km/h. However, the ROI can only be set to ROI 1 for straight roads, ROI 2 for right curves and ROI 4 for left curves. This will be solved in case 3 because of the implementation of the lane classifier. Finally, case 4 will use all three classifiers in the system. As a result, the best approximated ISP pipeline version can be chosen via the output of the classifiers, which will also change the sampling period and sensor-to-actuation delay.

In this section, each situation will be evaluated separately, which means that there is no transition between different situations. All the tested situations are already listed in Section 4.3.2.

### 5.2.2 Results and Analysis

Figure 5.1 shows the result of straight road. Since ROI 1 is fixed for these situations, all four cases have good performance. Figure 5.2 and 5.3 show the result of right and left turn respectively. Case 1 always fails in these situations because of its static ROI while case 2 fails in situation 13, 14, 20 and 21 since its ROI can't meet the need of dashed lines without the help of the lane classifier.

| Sit.No. | Classifiers used | ISP knobs | ROI knobs | Control knobs [h,tau](ms) | Speed |
|---|---|---|---|---|---|
| 1 | none | S0 | ROI1 | [25, 24.6] | 50 km/h |
| 2 | road classifier | S0 | change dynamically within {ROI1, ROI2, ROI4} | [35, 30.1] | 30 km/h<br>50 km/h |
| 3 | road + lane classifier | S0 | change dynamically within ROI 1-5 | [40, 35.6] | 30 km/h<br>50 km/h |
| 4 | road + lane +scene classifier | change dynamically | change dynamically within ROI 1-5 | change dynamically | 30 km/h<br>50 km/h |

Table 5.1: Considered Case for Static Situation-specific Test

For case 3 and 4, the vehicle is able to run stably. Thus, it can be concluded that the robustness of LKAS is improved.

On the other hand, the classifiers will result in downgraded control performance. For situations with straight road layout, case 1 has the smallest MAE among case 1-3. Each classifier will add 5.5 ms to sampling period and sensor-to-actuation delay, and with higher timings, QoC will decrease. As a result, case 3 is also outperformed by case 2 in curve situations except situation 15 and 16, which is due to additional sensor noise encountered in left turns. This can be solved by modelling sensor noise in a linear-quadratic gaussian (LQG) controller, which can be a research direction in future work.

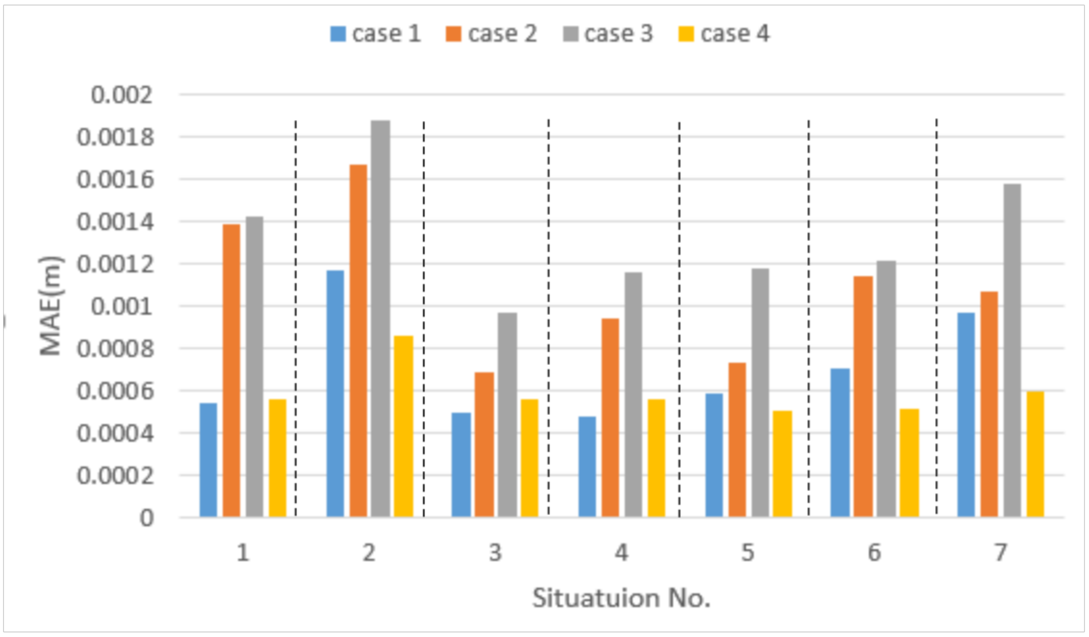

Figure 5.1: Result of Static Situation with Straight Road Layput

To overcome the runtime penalty brought by classifiers, approximated ISP pipeline can reduce the runtime of image processing. Case 4 uses three classifiers to determine the best ISP pipeline version for the current situation based on Table 4.2. As a result, case 4 performs better than case 3 in most situations except situation 15. This is due to the reduced image quality of approximated ISP pipeline have a bigger impact on QoC than smaller sampling period in this situation. Therefore QoC of case 4 is worse than case3. It is also apparent that situation 13, 14, 20, 21 have a high MAE. This is because of the fixed ROI for situations with two dashed lines. The vehicle will be settled more left or right to the middle of the lane in left or right curves, respectively, by doing this, the vehicle can avoid crashing. Although their QoCs in these situations are not good comparing

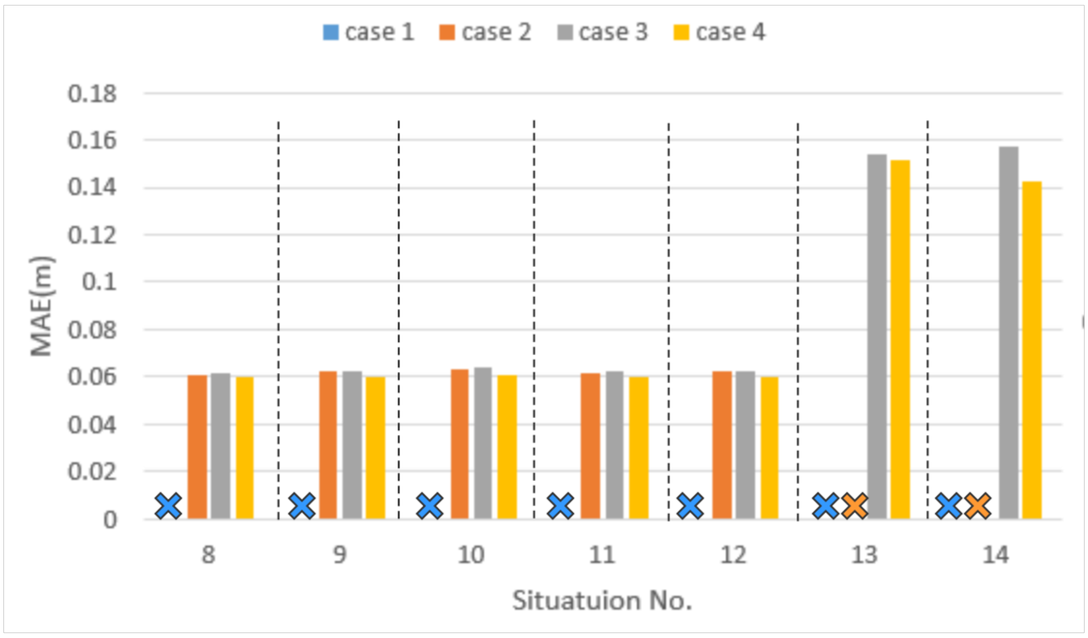

Figure 5.2: Result of Static Situation with Right Curve Layput

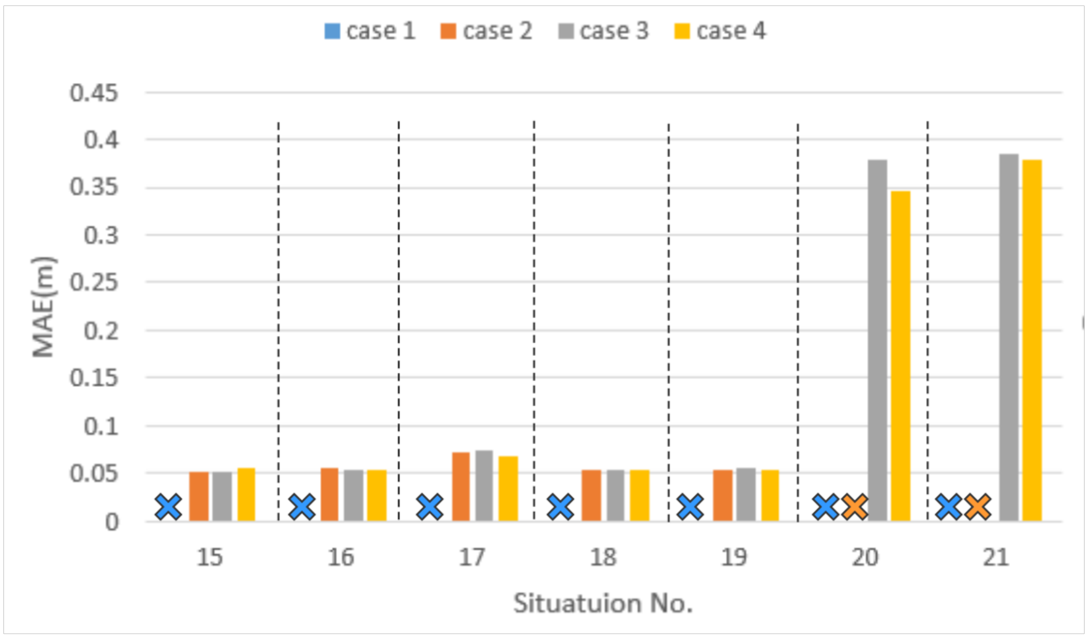

Figure 5.3: Result of Static Situation with Left Curve Layput

to others, the robustness is guaranteed. Comparing case 1 and case 4, the result shows that the classifiers can improve the robustness while the QoC will not decrease a lot. Thus the classifiers achieve its goal in static situations.

## 5.3 Analysis of Dynamic Switching between Situations

### 5.3.1 Experimental Setup

In real-world, the situations may change with the running of the vehicle. A world model is created for studying the dynamic switching between situations in Webots, as shown in Figure 5.4. Table 5.2 shows the situation of each sector in the world model. The vehicle will start from sector 1, and while it enters the next sector, LKAS will switch to the corresponding situation. This track contains dynamic road layout switch such as sector 1 to 2, lane type switches such as sector 5 to 6 and scene switch such as sector 8 to 9. Thus it can be considered as a concrete case study for evaluating the robustness and QoC of LKAS.

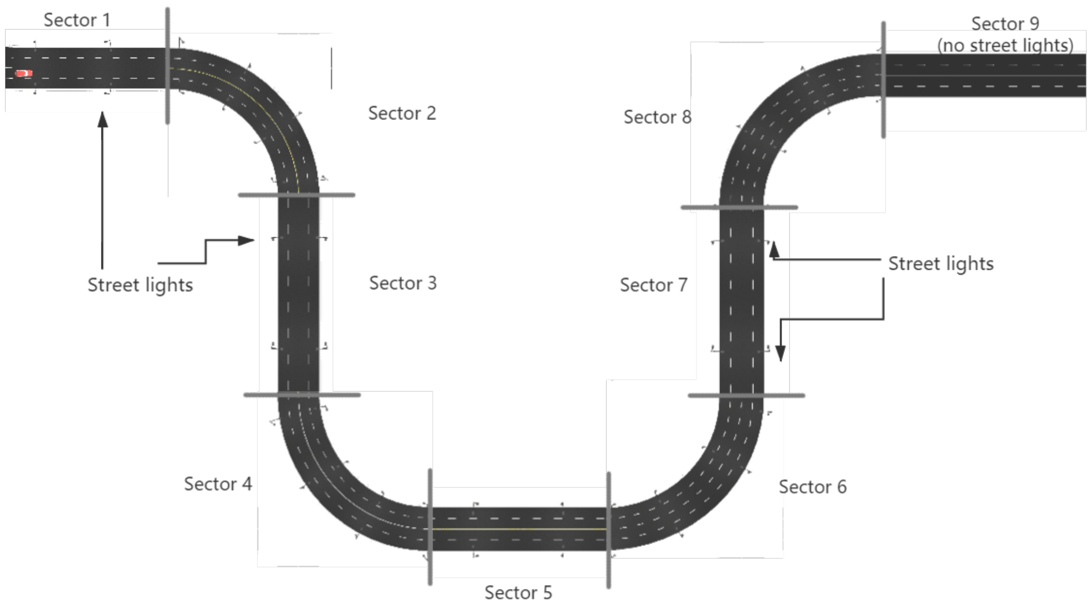

Figure 5.4: Model for Dynamic Switching between Situations

| Sector No. | Situation No. | Road layout | Left lane marker type | Scene |
|---|---|---|---|---|
| 1 | 5 | straight | white-continuous | night |
| 2 | 12 | right | yellow-continuous | night |
| 3 | 6 | straight | yellow-continuous | night |
| 4 | 18 | left | white-continuous | night |
| 5 | 6 | straight | yellow-continuous | night |
| 6 | 21 | left | white-dashed | night |
| 7 | 5 | straight | white-continuous | night |
| 8 | 14 | right | white-dashed | night |
| 9 | 7 | straight | white-continuous | dark |

Table 5.2: Situations for Dynamic Switching

### 5.3.2 Result and Analysis

Figure 5.5 shows the result of dynamic switching between situations. The results are all normalized to case 3. For case 1, the vehicle crashes when it is running from sector 1 to 2 because of the absence of dynamic ROI switch. Case 2 can run a bit further. However, the vehicle crashing while moving from sector 5 to 6. Case 3 and 4 can run through the whole track. This can illustrate the improved robustness of LKAS by using hardware- and situation-aware sensing.

On the other hand, the performance for LKAS with classifiers decreases. Case 3, which is used as the baseline, performs 55% and 22% worse than case 1 and case 2, respectively, on average. This comparison is only considering sectors without vehicle crash. For case 4, it performs 30% better than case 3 due to the implementation of the approximated ISP pipeline. Thus, the QoC of LKAS with hardware- and situation-aware sensing also improves while situations are dynamically switching.

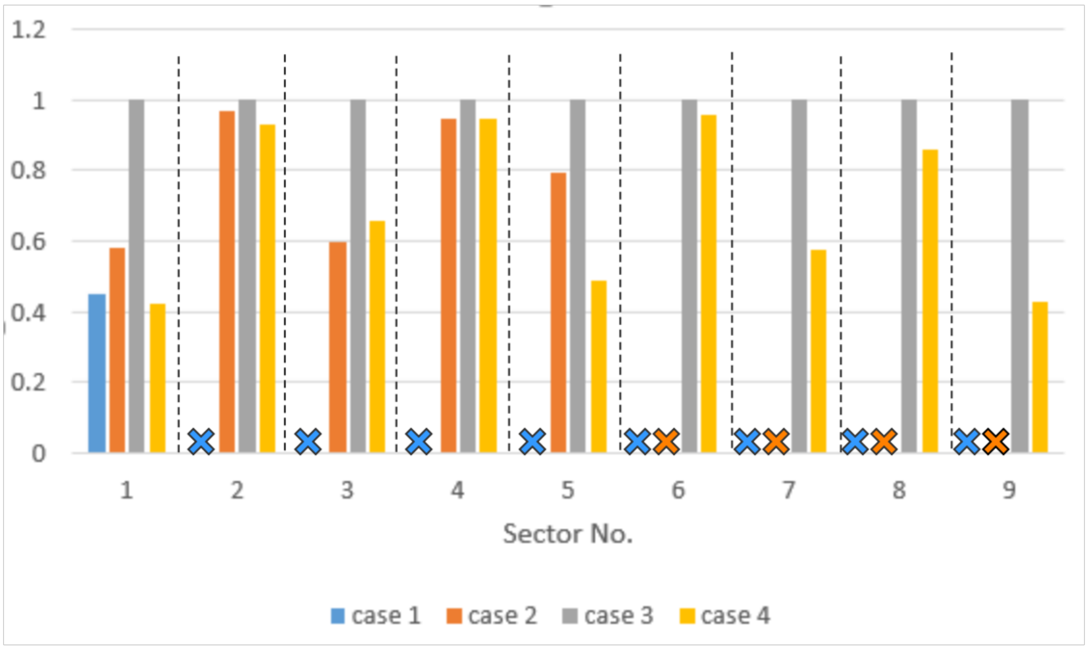

Figure 5.5: Result of Dynamically Switching Situation

## 5.4 Tuning Invocation Frequency of Classifiers

### 5.4.1 Experimental Setup

In Section 5.3, case 4 invokes all three classifiers in each frame, which brings a 16.5 ms runtime penalty. This penalty makes the sampling period and delay increased by 60% for ISP S0. Furthermore, for ISP pipeline versions skipping more stages, the time penalty even reaches 200%. Thus, an invocation frequency of these classifiers is considered as a solution to improve QoC of LKAS.

An evaluation window of 300 ms is considered as the invocation frequency. The reason for this is that the perception part of LKAS calculates the lateral deviation based on a 5.5 m look-ahead distance. Since the top vehicle speed in this project is 50 km/h, the current control decision can be valid for 400 ms. To prevent the instability of the system, a tighter window of 300 ms is chosen. Figure 5.6 shows the timeline of the invoked classifier. In every evaluation window, the first sampling period will invoke lane classifier, the in the second sampling period scene classifier will be invoked. For the rest sampling periods of this evaluation window, only road classifier will be invoked, because road layout changes more often in real-world and the robustness of LKAS is more sensitive to this change. This cycle will repeat every 300 ms.

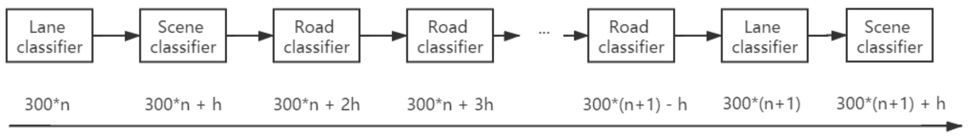

Figure 5.6: Timeline of Invoked Classifier

### 5.4.2 Result and Analysis

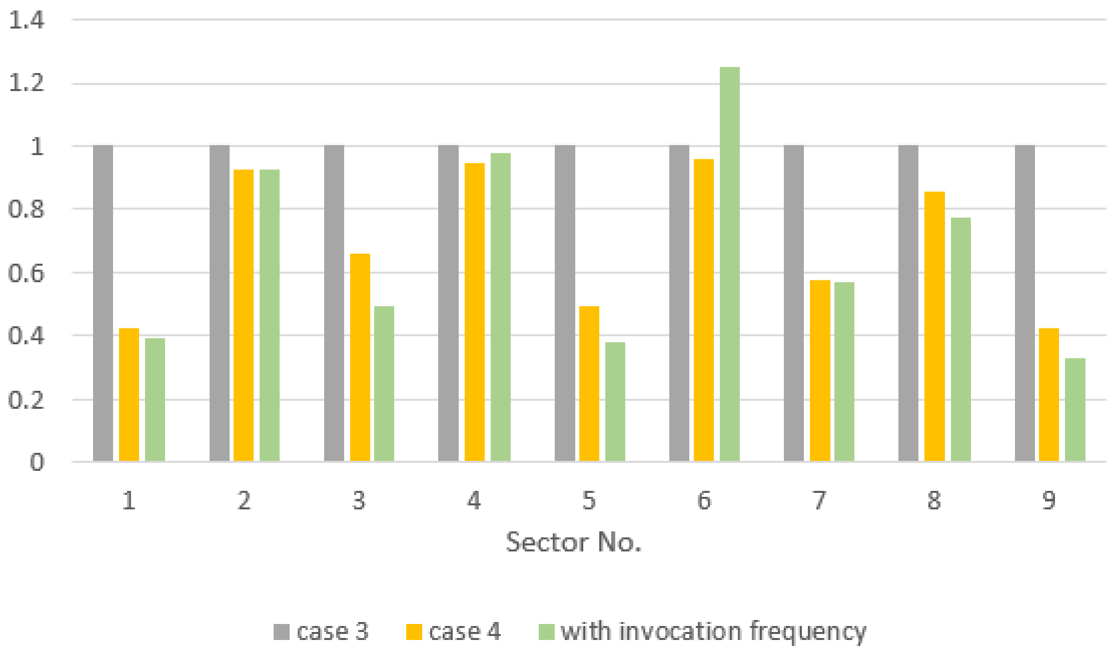

Figure 5.7: Result of Dynamically Switching Situation with Invocation Frequency

Figure 5.7 shows the result of dynamically switching situation with invocation frequency. For all the sectors with straight road and right curve layouts, the QoC is better than both case 3 and 4. Only in a left curve such as sector 4 and 6 the invocation frequency makes the QoC worse. This is due to the sensor noise brought by the right dashed line in a left turn. Introducing invocation frequency can improve the QoC by 32% and 3% comparing with case 3 and 4, respectively, on average. Thus, such a method to reduce the penalty brought by classifiers makes sense.

# Chapter 6

# Conclusions

## 6.1 Summary

This work focuses on improving the robustness and QoC of an LKAS system. First, two different types of approaches are compared. Although end-to-end CNN-based sensing methods have high accuracy, the traditional sensing method is finally chosen because of its fast speed on an embedded edge device. Along with the ISP pipeline and actuation part, these three parts make up an LKAS system.

We introduce hardware- and situation-aware sensing to this LKAS system. First, several situations are defined by three main features: road layouts, types of lane markers and types of scene. Three light-weight CNN situation classifier are trained and implemented into the LKAS system to identify each feature. These classifiers have a high accuracy of 96% and a relatively low runtime of 5.5 ms. Then we define 21 situations and test them to get their best configuration of approximated ISP pipeline version, the region of interest, sampling period, vehicle speed and sensor-to-actuation delay. After setting up such a look-up table, a dynamic runtime reconfiguration strategy is designed to ensure that LKAS can identify the current situation the vehicle is running under and then change to the corresponding configuration.

Using this new LKAS system, we test both static and dynamic situations. For the static situations, we see that although the time penalty brought by classifier have a negative impact on QoC, it can highly improve the robustness. Moreover, thanks to the approximated ISP pipeline, the QoC can also become better. For dynamic situations, we create a track in Webots, which includes several different situations changing with road layouts, type of lines and type of scene. It is obvious that comparing to the traditional LKAS without hardware- and situation-aware sensing, our approach can make the car successfully running through the whole track without crashing. Besides, QoC is also 30% better than LKAS with only two classifiers.

Finally, to further improve QoC, we design an invocation scheme to reduce the sampling period and sensor-to-actuation delay. An invocation frequency window of 300 ms can help LKAS to gain a 3% QoC improvement.

## 6.2 Future works

There are several future research directions. The first one is using a better simulator such as LGSVL can replace Webots as the server since more dynamic situation changes can be tested in it. Moreover, more features such as traffic and weather condition can also be taken into account while they are hard to realize in Webots.

Another direction is that one classifier which can identify different features at the same time can be developed. This can reduce the time penalty and simplify the system comparing using an invocation scheme.

# Bibliography

[1] NVIDIA AGX Xavier, Scalable AI Platform for Autonomous Driving. 2, 14

[2] Jamel Baili, Mehrez Marzougui, Ameur Sboui, Samer Lahouar, Mounir Hergli, J Subash Chandra Bose, and Kamel Besbes. Lane departure detection using image processing techniques. In *2017 2nd International Conference on Anti-Cyber Crimes (ICACC)*, pages 238–241. IEEE, 2017. 5

[3] Martin Becker, Sajid Mohamed, Karsten Albers, PP Chakrabarti, Samarjit Chakraborty, Pallab Dasgupta, Soumyajit Dey, and Ravindra Metta. Timing analysis of safety-critical automotive software: The autosafe tool flow. In *Asia-Pacific Software Engineering Conference (APSEC)*, pages 385–392. IEEE, 2015. 13

[4] Klaus Bengler, Klaus Dietmayer, Berthold Farber, Markus Maurer, Christoph Stiller, and Hermann Winner. Three decades of driver assistance systems: Review and future perspectives. *IEEE Intelligent transportation systems magazine*, 6(4):6–22, 2014. 1

[5] Mark Buckler, Suren Jayasuriya, and Adrian Sampson. Reconfiguring the imaging pipeline for computer vision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 975–984, 2017. 1, 3, 5, 6, 9, 10

[6] Vinay K Chippa, Debabrata Mohapatra, Anand Raghunathan, Kaushik Roy, and Srimat T Chakradhar. Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency. In *Design Automation Conference*, pages 555–560. IEEE, 2010. 2, 5, 6

[7] S. De, Y. Huang, S. Mohamed, D. Goswami, and H. Corporaal. Hardware- and situation-aware perception for robust closed-loop control systems. In *DATE*, 2021. 3

[8] S. De, S. Mohamed, K. Bimpisidis, D. Goswami, T. Basten, and H. Corporaal. Approximation trade offs in an image-based control system. In *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1680–1685, 2020. 6, 7

[9] S. De, S. Mohamed, D. Goswami, and H. Corporaal. Approximation-aware design of an image-based control system. *IEEE Access*, 8:174568–174586, 2020. 1, 2, 6, 7

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 21

[11] Alexandru Gurghian, Tejaswi Koduri, Smita V Bailur, Kyle J Carey, and Vidya N Murali. Deeplanes: End-to-end lane position estimation using deep neural networksa. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 38–45, 2016. 6

[12] J. Han and M. Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *2013 18th IEEE European Test Symposium (ETS)*, pages 1–6, 2013. 5

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 21

[14] Ku He, Andreas Gerstlauer, and Michael Orshansky. Controlled timing-error acceptance for low energy idct design. In *2011 Design, Automation & Test in Europe*, pages 1–6. IEEE, 2011. 5, 6

[15] Haomiao Jiang, Qiyuan Tian, Joyce Farrell, and Brian A Wandell. Learning the image processing pipeline. *IEEE Transactions on Image Processing*, 26(10):5032–5042, 2017. 6

[16] Takeo Kanade, Chuck Thorpe, and William Whittaker. Autonomous land vehicle project at cmu. In *Proceedings of the 1986 ACM fourteenth annual conference on Computer science*, pages 71–80, 1986. 1

[17] Wen-Chung Kao, Sheng-Hong Wang, Lien-Yang Chen, and Sheng-Yuan Lin. Design considerations of color image processing pipeline for digital cameras. *IEEE Transactions on Consumer Electronics*, 52(4):1144–1152, 2006. 5

[18] Jihun Kim, Jonghong Kim, Gil-Jin Jang, and Minho Lee. Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection. *Neural Networks*, 87:109–121, 2017. 6

[19] Jihun Kim and Minho Lee. Robust lane detection based on convolutional neural network and random sample consensus. In *International conference on neural information processing*, pages 454–461. Springer, 2014. 6

[20] ZuWhan Kim. Robust lane detection and tracking in challenging scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):16–26, 2008. 6

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[22] S. Lee, J. Kim, J. S. Yoon, S. Shin, O. Bailo, N. Kim, T. Lee, H. S. Hong, S. Han, and I. S. Kweon. Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1965–1973, 2017. 2

[23] Chan Yee Low, Hairi Zamzuri, and Saiful Amri Mazlan. Simple robust road lane detection algorithm. In *2014 5th International Conference on Intelligent and Advanced Systems (ICIAS)*, pages 1–4. IEEE, 2014. 5

[24] Olivier Michel. Cyberbotics ltd. webots$^{\text{TM}}$: professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):5, 2004. 8

[25] Sajid Mohamed, Asad Ullah Awan, Dip Goswami, and Twan Basten. Designing image-based control systems considering workload variations. In *58th IEEE Conference on Decision and Control (CDC)*, 2019. 13

[26] Sajid Mohamed, Sayandip De, Konstantinos Bimpisidis, Vishak Nathan, Dip Goswami, Henk Corporaal, and Twan Basten. Imacs: a framework for performance evaluation of image approximation in a closed-loop system. In *2019 8th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–4. IEEE, 2019. 1, 2, 3, 6, 7

[27] Sajid Mohamed, Dip Goswami, Vishak Nathan, Raghu Rajappa, and Twan Basten. A scenario-and platform-aware design flow for image-based control systems. *Microprocessors and Microsystems*, 75:103037, 2020. 13

[28] Sajid Mohamed, Nilay Saraf, Daniele Bernardini, Dip Goswami, Twan Basten, and Alberto Bemporad. Adaptive predictive control for pipelined multiprocessor image-based control systems considering workload variations. In *59th IEEE Conference on Decision and Control (CDC)*, 2020. 13

[29] Sajid Mohamed, Diqing Zhu, Dip Goswami, and Twan Basten. Optimising quality-of-control for data-intensive multiprocessor image-based control systems considering workload variations. In *21st Euromicro Conference on Digital System Design (DSD)*, pages 320–327, 2018. 13

[30] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 286–291. IEEE, 2018. 2, 6

[31] Rajeev Ramanath, Wesley E Snyder, Youngjun Yoo, and Mark S Drew. Color image processing pipeline. *IEEE Signal Processing Magazine*, 22(1):34–43, 2005. 5

[32] Eli Schwartz, Raja Giryes, and Alex M Bronstein. Deepisp: Toward learning an end-to-end image processing pipeline. *IEEE Transactions on Image Processing*, 28(2):912–923, 2018. 6

[33] C. Turner. Next-generation automotive image processing with arm mali-c71, 2017. 1

[34] M. Williams. Prometheus-the european research programme for optimising the road transport system in europe. In *IEE Colloquium on Driver Information*, pages 1/1–1/9, 1988. 1

[35] Yasin Yenİaydin and Klaus Werner Schmidt. A lane detection algorithm based on reliable lane markings. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE, 2018. 5

[36] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020. 1

[37] Majid Zamani, Soumyajit Dey, Sajid Mohamed, Pallab Dasgupta, and Manuel Mazo. Scheduling of controllers' update-rates for residual bandwidth utilization. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 85–101. Springer, 2016. 13

[38] Shengyan Zhou, Yanhua Jiang, Junqiang Xi, Jianwei Gong, Guangming Xiong, and Huiyan Chen. A novel lane detection based on geometrical model and gabor filter. In *2010 IEEE Intelligent Vehicles Symposium*, pages 59–64. IEEE, 2010. 5