The Eyes

Contents

[1 Introduction 2](#_Toc26456507)

[2 Objectives 2](#_Toc26456508)

[2.1 General Objective 2](#_Toc26456509)

[2.2 Specific Objectives 2](#_Toc26456510)

[3 State of the Art 2](#_Toc26456511)

[3.1 Different approaches for path recognition 2](#_Toc26456512)

[3.2 Types of QR code readers 4](#_Toc26456513)

[3.3 Simulation mechanisms 4](#_Toc26456514)

[4 Equipment Chosen 4](#_Toc26456515)

[4.1 Hardware 4](#_Toc26456516)

[4.2 Software 5](#_Toc26456517)

[4.2.1 Main Programs 5](#_Toc26456518)

[4.2.2 Others 5](#_Toc26456519)

[5 Procedure 6](#_Toc26456520)

[5.1 Analyzing the autonomous car 6](#_Toc26456521)

[5.1.1 The onboard camera and the webcam 6](#_Toc26456522)

[5.1.2 The steering system driven by data using Arduino 6](#_Toc26456523)

[5.1.3 Other components 6](#_Toc26456524)

[5.2 Coding the path recognition algorithm 6](#_Toc26456525)

[5.3 Coding the QR Readers 6](#_Toc26456526)

[5.4 Simulation and testing 6](#_Toc26456527)

[6 Results 6](#_Toc26456528)

[6.1 Simulation results 6](#_Toc26456529)

[6.2 Test results 6](#_Toc26456530)

[7 Conclusions 6](#_Toc26456531)

# Introduction

# Objectives

## General Objective

1. To drive autonomously the Audi car along the racing track.

## Specific Objectives

1. Detecting Lanes.
2. Detecting QR Codes.
3. Designing one racing tracks.
4. Running simulations on the racing tracks.
5. Testing the code on the Audi car.

# State of the Art

## Different approaches for path recognition

Example 1:

https://www.youtube.com/watch?v=a6pDdS6sY2E

Example 2 (Long):

https://www.youtube.com/watch?v=VyLihutdsPk

Example 2:

https://www.youtube.com/watch?v=0se6\_UPNWVc

Fun to watch:

https://www.youtube.com/watch?v=IzvlqCEYjg4

https://www.youtube.com/watch?v=X8QN-qY7uIo

https://www.youtube.com/watch?v=gWK9x5Xs\_TI

https://www.youtube.com/watch?v=G2VaJvNNp4k

https://www.youtube.com/watch?v=JmxDIuCIIcg

https://www.youtube.com/watch?v=ipXQFcAeovk

Recommendation: Record a video from the onboard camera of the car going through either the simulatin or a real track. WIll use this video to try out the codes as in the video.

What color are the lanes Vs what color is the background?

Camera Lens distortion? Useful to warp the image. OpenCV has camera calibration and Undistort functions.

Perspective transformation. OpenCV can transform the lanes perspective.

Using Color spaces to detet lanes or using edge detections.

Threshold both the color channels and the edge frames. Find the Hot pixels.

Histogram is helpful for curve fitting. Guessing where the lanes start on the binarized image.

Create boxes that follow along the lines Then use a Polinomial fit to find the curvature of thhe lane.

HoughLanesP code from OpenCV seems to be an useful function to detect lines given some parameters.

Once the lanes are found, it is necesary to determine by geometry some directional vectors.

We can either do calculations ahead in time and predict curves or go through them on the run by running a code to keep the image centered.

1) Get the sample onboard camera video or simulation video.

2) Crop in the area of interest

3) apply flters to reduce noise if necessary.

4) Create functions to detct lanes using OpenCV library. (There are functions that already do this like: HoughLanesP, Edge Detection Filters, )

5) Once the lanes are found, determine the geometry of them. Draw directional vectors.

6) By doing math, determine the focus point of the curve. Convert the camera perspective of the curve to an estmate geomety of the real curve.

7) Do geometry conversion and turn each wheel to match the curve focus in an Ackerman Steering System.

8) Pray that it works.

- Write functions for:

following a lane. Keeping the lines convergent at equivalent degrees of inclination and centered with the vertical axis.

detecting a curve. Either estimate it in advance or run through it live.

entering into a curve. Small deviation from following the lanes to match the curve on time.

matching the curvature of the curve. The focus of the Ackerman Steering has to match the focus of the curve.

finding the end of the curve.

exiting the curve. Small deviation from following the curve to match the straight lane again.

## Types of QR codes readers

QR Team could write more info here. Kinds of different QR codes variations that exist.

## Simulation mechanisms

Talk about different ways of simulation.

# Equipment Chosen

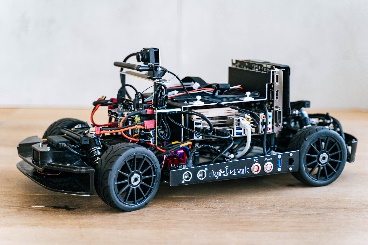
## Hardware

1. ADAS Audi model Q3.

<https://www.digitalwerk.net/adas-modellauto/>

**drive**

* Brushless motor for the drive
* Cruise control for forward and reverse, as well as brakes in both directions.
* Digital Servomotor with high power at low duty cycle to ensure a precise and fast steering.



**power supply**

The vehicle has two separate energy circuits.

* In order to enable working on the vehicle (measuring technology and computer) without any problems there is a power supply 6-24V.
* Measuring and control computer:  
  Due to the higher power consumption, a 6-cell battery with 22.2V and 5.200mAh is used and can be operated for one hour under full load.
* The engine and cruise control is powered by a 2-cell battery with 7.4V.

We also need to mention information about the onboard camera.

1. Generic usb webcams.

We are using some plug and play usb cameras from the LAS.

1. Laptops.

## Software

### Main Programs

1. Ubuntu v16.04

The ADAS car works with ROS, and ROS works with Linux. Therefore, we are using the last stable Ubuntu version that has been used in the Laboratory for Autonomous Systems (LAS L001).

1. ROS Kinectic

Compatible with Ubuntu 16.04

1. Solidworks

Rending solids and creating 3D racing tracks.

1. Carla/Gazebo

Physics simulations.

1. Visual Studio Code

Scripting program to write code in Python, while using OpenCV libraries and connecting to ROS.

1. Pycharm

Used for…

### Others

1. Python

The ROS environment and the OpenCV work with some programming languages, especially with C++ and Python. The team opted out to try a new programming language we all haven’t tried before so we chosed Python.

1. OpenCV

It is a popular widely used open source library for image processing. Written natively in C++, works with C++, Python, Java and Matlab.

# Procedure

## Coding the path recognition algorithm

Briefly describe the pat recognition algorithm

Talk about ROS node chatter and talker. Subscribing to ROS

Mention the Ackerman steering system driven

Talk briefly about the driving control system by data using Arduino

## Coding the QR Readers

Briefly describe the QR code.

## Simulation and testing

We created some racing tracks in Solidworks, exported mesh files and ran simulations in Carla with those files.

# Results

## Simulation results

Having problems with Gazebo. Models are not working because the hardware support of our PC are incompatible or insufficient. Only 1 computer managed to work with Gazebo but even then some models crashed, even if we followed the tutorials.

We decided to switch to a better simulation program recommended by Simon, Carla. But we also are required to have computers with much better specifications.

Gazebo doesn’t have proper rendering, therefore we decided to also use Solidworks to create 3D mesh files. The racing track will be created in Solidworks and then exported into Gazebo or Carla for physics simulation.

## Test results

Write some expected results from the test Results

The car successfully went through the racing track 9 out of 10 times with an average completion time of 30s.

# Conclusions

1. Write some expected conclusions.