Middle East Technical University

Department of
Electrical and Electronics Engineering

EE493 ENGINEERING DESIGN I

Car Chasing Robot
Conceptual Design Report

Supervisor: Assoc. Prof. Emre Özkan
METU EE / C-112

Project Start: 4/10/2018
Project End: 26/5/2019
Project Budget: $450

Company Name : Duayenler Ltd. Şti.

Members	Title	ID	Phone
Sarper Sertel	Electronics Engineer	2094449	0542 515 6039
Enes Taştan	Hardware Design Engineer	2068989	0543 683 4336
Erdem Tuna	Embedded Systems Engineer	2617419	0535 256 3320
Halil Temurtaş	Control Engineer	2094522	0531 632 2194
İlker Sağlık	Software Engineer	2094423	0541 722 9573

December 26, 2018

This page intentionally left blank.

Contents

1 Executive Summary

2 Introduction

DUAYENLER is established with the aim of developing autonomous car technologies for near future. To serve that purpose, Car Chasing Project is initiated by the company. The project can be summarized as a vehicle that can autonomously follow a path and detect the other surrounding vehicle as well as communicating them to have a reliable driving environment. With this project, the company aims to accomplish the following objectives:

1. Sensing the environment and other vehicles on the roads
2. Automatic adaptive lane detection
3. Self driving
4. Autonomous wireless communication with surrounding counterparts

A considerable amount of effort and work force has been put on the project to fulfill the required objectives. So far, the team has figured out several important steps towards the realization of the project.To start with, the wireless communication between the vehicles is modeled and implemented. A reliable communication environment is established using Wi-Fi protocol. Currently, the vehicles can communicate with each others by means of associated handshake protocol messages in a race scenario. Secondly, computer vision algorithms are developed and implemented as a solution to lane detection problem. The algorithms are developed based on open source computer vision library OpenCV. To obtain a direction predicting results, color thresholding, edge detection, hough transform algorithms are used respectively. Furthermore, the communication between image processing platform and microcontrollers for motor driving is constructed. It is the essential part of solving the self driving problem. On the mechanical part, different motor&wheel combinations are tested to obtain the best performance. To test the computer vision on board, a prototype vehicle is assembled and necessary equipment is mounted on it. Currently, the team is working on the improvement of computer vision algorithms.

In this report, the company provides technical details about the implemented solutions, other possible solution alternatives with objective comparisons as well as a clear action plan showing the necessary further steps for realization of the project. The emphasis on this report is primarily put on the detailed analysis of proposed solutions, supported with relevant test results in both system and subsystem level. In addition, future plans including new test designs for current solutions as well as for other alternatives, the action plan in case of unexpected outcomes by clearly specifying the responsibilities of each member in the team.

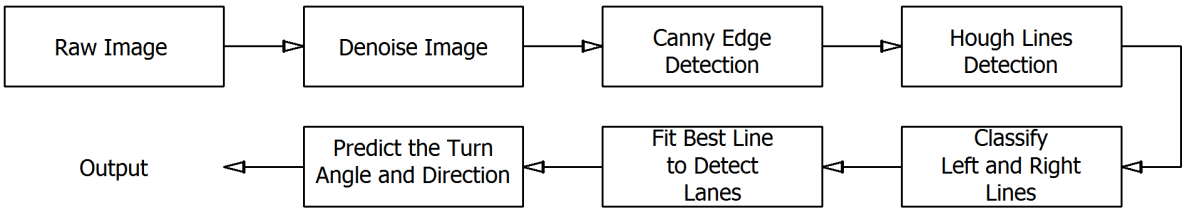

Figure 1: The Proposed Algorithm for the Lane Detection.

3 Solutions

3.1 Description of individual sub-systems

3.1.1 Sensing System

Sensing system has two main subsystems which are "Lane Detection Subsystem" and "Vehicle Detection Subsystem".

3.1.1.1 Lane Detection Subsystem

The subsystem basically detects the lane. This subsystem uses OpenCV libraries for processing camera frames. The input is captured from a Raspberry Pi camera that is mounted to the vehicle. The captured frame is firstly, preprocessed by a denoising filter. Then, the edges in the frame is detected by Canny Edge Detector algorithm. The output of Canny is a binary image filled with ones and zeros. The Hough Lines technique probabilistically extracts possible lines in a given binary image. Thus, possible line constituting points are obtained from Hough Lines function. Next, points of the lines are classified as left or right borders of the lane. The elimination of the wrong points are done concurrently with the classification. Then, filtered points are fitted in two separate lines to create left and right borders of the lane. As the lane borders are found, the next and the last step is to determine the direction of the vehicle. The output of this subsystem is a turning angle and a direction.

3.1.1.2 Vehicle Detection Subsystem

The detection of the opponent can be implemented using distance sensors. In order to stop the robot when it catches the opponent, or it is caught, a sensor must be placed at the front and the back. The most common distance sensors are ultrasonic, infrared and laser sensors.

Ultrasonic sensors have acceptable range. They send a sound wave and take back the echo, then give a PWM voltage related to the distance. However, using ultrasonic sensors can cause problems if the opponent also uses ultrasonic sensors; because of the interference. Besides, when measurement is angled, measuring is failing.

Infrared sensors can be another approach to this problem, they may provide better accuracy. Moreover, Laser distance measurement can be applied to solve this problem. They have the best accuracy, but their price is the highest among the rest.

3.1.2 Computation System

Functionality of computation subsystem will be presented in this section.

3.1.2.1 Data Processing Subsystem

This unit is the main algorithm application level. Data from sensor will be aggregated in this unit and will be pre-processed. Then, processed data will be output to the controller unit. Processing will mainly be done using Arduino and Raspberry Pi (if used). Suitable algorithms will be developed to realize desired operations with the controller unit

3.1.2.2 Controller Subsystem

Mathematical models will be implemented in this unit to realize controllers such as P, PI, PID. The output of the controller unit will be sent to driving subsystem. This output will be the ultimate decision to realize a desired operation according to the data sent from sensor.

3.1.3 Communication System

3.1.3.1 Internal Communication Subsystem

3.1.3.2 External Communication Subsystem

3.1.4 Driving System

This unit takes an input from the controller unit. That input involves information about what the new orientation should be. Driving subsystem, then, takes action to move vehicle according to the required direction with maximum possible speed. This subsystem consists of two units: Direction and Speed.

3.1.4.1 Direction Subsystem

Direction unit is responsible for the orientation of the vehicle. It stores the last required orientation and the new one coming from the controller. After that, it tries to make the orientation as close as new one. Both data can be represented as vectors.The angle between those two vector is tried to be minimized by the controller. Before moving on to the operation, note that the angle can be used as a measure of the error that the direction unit have. The less the angle the more correctly operates the direction unit.

Depending on the configuration of the wheels, exact control of the vehicle might vary. However, there are certain methods to accomplish orientation. The vehicle will definitely have two wheels or palettes that will be driven by two separate DC motors. That configuration allows differential drive method to orient the vehicle. PWM values of the motors can be adjusted such that the speed difference between them results in a turn as much as desired angle. The exact difference values on the PWM values depends on the specs of the used motors and voltage sources.

Two different H-bridge motor drivers are proposed to be used to drive DC motors: L298N and L293D. Both can drive two motors separately with one IC. However, maximum current rating of the former one is larger being 2A while L293D can supply 0.6A per channel.

As in the case of another configuration that involves one or two servo motors to control the directions of the front wheels. This configuration is more robust compared to ball caster utilization. However, there are more motors to control and it requires more complicated differential drive algorithms involving both DC motor differential and servo PWM to orient the front wheels.

3.1.4.2 Speed Unit

This unit acts as a complementary module for direction unit. It will act as a state machine. In one state, the unit will try to increase the speed of the vehicle by making overall increase in both PWM values of DC motors. The feedback of this system will be the cost function mentioned in driving unit. If that cost exceeds a specified level, unit goes to another state in which the unit will decrease the overall speed to allow direction unit to operate more correctly. In short, this unit tries to compensate the error of the direction unit by changing the overall speed of the vehicle.

3.1.5 Structure Subsystem

This part contains chassis and PCB sections of the robot.

3.1.5.1 Chassis Subsystem

Main purposes of this section are protection of the critical elements of the robot and holding components together. The most important part of this section is weight distribution. The chassis is supposed to be light and strong because of the competition purposes. However, it should balance the robot to be able to handle with turns.

3.1.5.2 Printed Circuit Board Subsystem

The main role of this part is decreasing connection mass and increase vibration strength of the robot against disturbances. Also, this section increases rigidity of the whole system.

3.1.6 Motion System

Motion of the system is detailed in this section.

3.1.6.1 Wheels Subsystem

There are possible solution for wheel placement on the chassis, and several wheel types. Some wheels are designed for better gripping on different surfaces. To avoids obstacles on the path, gripping of the wheel is an important concept. Some wheel types are ball caster, toy car wheel and palette. Besides, wheel placement and the wheel number

should be combined with the wheel type choice.

One of the possible wheel placement is 2+1 combination. This combination can be assembled by placing 2 car wheels (with motors) to the back and the one boll caster to the front or vice versa. These configurations provide easy implementation and fairly reliable handling on the path. However, for certain obstacles may significantly disturb vehicles balance in this configuration.

Another combination is palette system. This system is used in real world where robust vehicles are needed. Similarly, this configuration can help handling obstacle in the path, but it costs for harder implementation and driving.

Last implementation is 2+2 configuration. In this configuration 2 wheels can be placed at the back and the rest at the front by placing motors to back wheels. To ease turning of the vehicle, front wheels can be controlled with a servo motor as back wheels operate in the differential drive mode. This combination may provide both enhanced grip and reliable operation.

3.1.6.2 Motors Subsystem

Motors are one of the most important physical components of the project. There are possible motor types in the market.

One of the widely used motor type is brushed DC motors. Such motors might be implemented with gears. Gears are utilized to adjust torque and RPM of the motor, which is very suitable for a racing vehicle's needs.

Another option is brushless DC motors. Brushless DC motors do not use brushes. This results in high torque. Brushless motors are more suitable for high RPM required areas such as CD drivers and drones.

Last option is servo motors. Servo motors are high-torque motors that can turn in an desired angle. Servos can be utilized in the direction of the vehicle on the front wheels. By using this solution, turning radius can be decrease significantly.

3.2 System & Subsystem Level Requirements

3.2.1 Sensing System Requirements

- The system should detect the sides of the road.
- The system should not be effected from external disturbances.
- The system should detect the opponent vehicle.

3.2.1.1 Lane Detection Subsystem Requirements

- The subsystem should be able to detect only the shades of green color
- The subsystem should be able to detect edges in the camera frame in any light condition
- The subsystem should be able to tell differences between disturbances and lane
- The subsystem should be able to interpret the middle of the lane if both sides are present at the frame

3.2.1.2 Vehicle Detection Subsystem Requirements

- The subsystem should detect the opponent to be caught with in a 5 cm
- The subsystem should detect the chasing opponent if it reaches from back with in a 5 cm

3.2.2 Computation System Requirements

- The system should be able to produce middle line to follow
- The system should be able to control the robot

3.2.2.1 Data Processing Subsystem Requirements

- The subsystem should be able to analyse data produced by sensing system
- The subsystem should be able to produce the angle information required by the controller subsystem
- The subsystem should be able to work on Raspberry Pi
- The subsystem should be able to process one frame at most in 100 milliseconds

3.2.2.2 PID Controller Subsystem Requirements

- The subsystem should be able to control the motors
- The subsystem should be able to react the external disturbances

3.2.3 Communication System Requirements

- The subsystem should ensure safe internal communication
- The subsystem should ensure safe external communication

3.2.3.1 Internal Communication Subsystem Requirements

- The microcontrollers should be able to communicate with each other via serial communication
- The internal communication speed should be compatible with the processing speed of the lane detection subsystem

3.2.3.2 External Communication Subsystem Requirements

- The subsystem should be able to communicate with the opponent via Wi-fi protocol
- The subsystem should be able to execute handshake protocol

3.2.4 Driving System Requirements

- The subsystem should control motion subsystem according to output of the computation system

3.2.4.1 Speed Subsystem Requirements

- The subsystem should decrease the vehicle speed at the narrow lane
- The subsystem should increase the vehicle speed at the wide lane
- The subsystem should decrease the vehicle speed at the extreme disturbance

3.2.4.2 Direction Subsystem Requirements

- The subsystem should drive the motors according to computation system outputs
- The system should ensure that the vehicle follows the lane

3.2.5 Motion System Requirements

- The system should ensure that the vehicle can drive itself with enough power

3.2.5.1 Wheels Subsystem Requirements

- The subsystem should ensure that the wheels can grip lane without slipping in all conditions

3.2.5.2 Motors Subsystem Requirements

- The subsystem should ensure that the motors can supply enough torque to accelerate the vehicle
- The subsystem should ensure that the motors can execute driving system outputs without deviation

3.2.6 Structure System Requirements

- The system should ensure that structure is robust for external effects
- The system should ensure that structure is balanced to increase handling

3.2.6.1 Chasis Subsystem Requirements

- The subsystem should ensure that the chassis is rigid
- The subsystem should ensure that the chassis have enough space for components
- The subsystem should ensure that the chassis can provide low center of mass

3.2.6.2 PCB Subsystem Requirements

- The subsystem should ensure that all the electronic devices are placed on PCB
- The subsystem should ensure that the components are not connected via loose cable

3.3 Solution for each subsystem and relevant algorithms

The applied algorithms, solution steps and solution methodology are presented in this section in a detailed manner.

3.3.1 Lane Detection Subsystem

The first processing on the captured frame is denoising by blurring. The blurring filter is a GaussianBlur of (3x3) matrix with zero variance both in x and y directions. Next, the filtered image is processed by Canny Edge Detector. This process eliminates all pixels except those are constituting an edge. Edge pixels are usually formed when there is a transition from one object to other.

Then those detected pixels are input to Hough Lines function. This function outputs two pixels points on the image that form a line. The output of the function is tens of such line points. The accuracy of the function can be adjusted by playing with the input parameters. The function basically draws all lines that might go through a point in the free space (in polar coordinates) and intersects those lines. If the origin points of the intersected lines are within a specified bound, then the line is defined and it can be expanded further. If found points on a line don't exceed a threshold, that line is discarded. The adjustment of such parameters are done with trial and error approach. An important note is that this function is probabilistic, meaning that even if the input frame is never changed, the output line points would be close to previous point but not the same. An example call for this function together with its parameters is given *Script 1.*

Then next step is to classify the line points as left or right. The lines are firstly eliminated according to their slopes. If slope of a line is not in invalid slope region of ± 0.005, then it is a valid line. This process is done to get rid of unnecessary low sloped

```
HoughLinesP(input=img_threshed, output=line,
        rho=1, theta=CV_PI / 180, threshold=15,
        minLineLength=30, maxLineGap=40);
```

Script 1: Hough Lines Function with its Parameters

```
std::vector<cv::Vec4i> lines // like a 4x4 matrix
double slope_thresh = 0.005; // absolute threshold slope
for (lines_points) {}
        ini = Point(x1, y1);
        fini = Point(x2, y2);
                line_slope = fini/ini;
        if(abs(line_slope) >= slope_thresh) line_is_valid;
                else line_is_not_valid;

for (lines_is_valid)
        if(x1>320 && x2>320)
                it_is_right_line;
        else if(x1<320 && x2<320)
                it_is_left_line;
        else discard_the_line;
```

Script 2: The Algorithm to Classify the Lane Lines as Right or Left

lines. After elimination, line points set must be determined as left or right. At this point, this classification is done according to double checking. the center of the image, that is 320th vertical pixel. If initial and final horizontal points of the image is in the same half, the line belongs to that half. This method works nicely in most regions of the path. However, it is a bit error prone in case of a sharp turning angle or losing one of the lanes . This algorithm will be improved to be more robust.

After finding the all left and right line points, actual left and right lines are constructed by applying least square method to the both point sets. The result of this method is 4 points (x1,y1,x2,y2) to refer the left lane line and 4 points to refer the right lane line.

The last process on the image is predicting the turn angle and direction. The prediction of direction is done by comparing the slopes of the left and the right lines. The turning is to be made to the side of the lane line having less slope. The angle is determined by computing the angle between the normal line of the current direction and guide line. Guide line is constructed with the average of the middle points of the right and the left lines and current point of the vehicle. Vehicle's current point is assumed to be in the middle of the beginning of the image. Thus, the beginning point of the guideline is the current position of the vehicle and the endpoint is the target point to be reached. The output of this processing is shown in *Figure 2*.

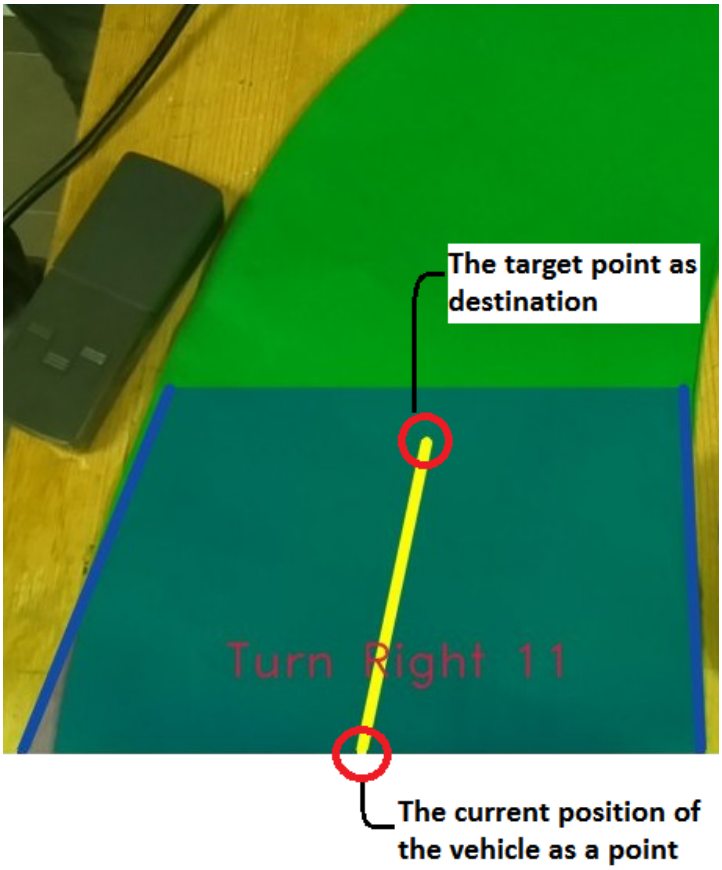

Figure 2: The Prediction of Turn Angle and Direction.