YANGON TECHNOLOGICAL UNIVERSITY
Department of Mechatronics Engineering

Integrated Design Project Report
Group No.(4)

Real-Time ROS-based Architecture for Autonomous Driving with Computer Vision

September 13th, 2019

Table of Contents

Abstract **Autonomous driving or self-driving is the ability of a vehicle to drive itself without human input. For vehicles to be able to drive by themselves, they need to understand their surrounding environments to follow the path, avoid obstacles, pause at stop signs and traffic lights. To achieve this, the vehicle uses mechanical and electronic parts, sensors, actuators, and on-board computer. The first task is robust navigation, which is based on system vision to acquire RGB images of the road for more advanced processing. Cameras are a crucial exteroceptive sensor for autonomous vehicles. They can be used for multiple purposes such as visual navigation and lane line detection since they can provide appearance information about the environment. And the second is the controlling of the vehicle's position, speed and direction. In this project, the reason and procedure for detecting lane lines in an image using computer vision techniques have been discussed. It also aims to demonstrate how ROS could be used to develop autonomous driving software by analysing autonomous driving problems, examining existing solutions and developing a prototype vehicle using ROS.**

---

1. Introduction

Automated vehicles are technological development in the field of automobiles. An autonomous car can sense their local environments, classify different kinds of objects that they detect, and can interpret sensory information to identify appropriate navigation paths whilst obeying transportation rules. Autonomous vehicles are challenged to navigation issues in an uncertain environment and latest artificial intelligence and computer vision techniques offer potential solutions for autonomous vehicles navigation in an unstructured environment, scene analysis, classification, etc. However, detecting lane lines on the road is still a difficult task for intelligent vehicles. It is essentially due to the huge amount of data that should be processed in real time. The algorithms should recover the road state and uncertainties issues, vehicle vibration, noises etc. To overcome the constraints, new methods should be fused with the traditional image processing algorithms to improve them and to better target the region of interest and to minimize the computational cost for the real-time applications.

This project focuses on the various image processing techniques to detect lane lines on the test environment and introduces a communication framework called Robot

Operating System (ROS). The organization of the project is as follows: the first part is dedicated to computer vision for real-time detection of lane lines. The second part focuses on the control system and kinematics modeling of the differential-drive mobile robot. In the third part, ROS is introduced and the following parts focus on design and experimentation of the real hardware robot and the project is concluded.

2. Computer Vision

There are many ways that can be used to achieve the purpose of perception in the self driving. One such method is computer vision. Computer vision's method use the camera as the sensing device, the frames as the inputs and process each and every frame got from the camera and give out the output. Computer vision method is relatively familiar and easier for the users and the customers to understand the mechanisms going on so it is the first step in perception step in self driving. The output (set point) gotten from this process can be used for the input in the control step.

2.1 Camera Calibration

Cameras capture the image by mapping the 3D real world objects onto the 2D image plane. The capture images are not precise and distorted in most of the cases. Ideal pinhole camera model uses *camera matrix* to transform 3D object points into 2D image points. Real camera uses lenses which can form images faster but they introduce distortion. Light rays tend to bend a little at the edges of the lens and this creates the distortion in images. There are two types of distortions called radial distortion and tangential distortion. Radial distortion make images to be slightly bent or curved at the edges and is the most common distortion. Tangential distortion happen when image plane and lens are not parallel and this makes the image longer or tilted. *Distortion coefficients* are used to undistort the distorted images.

2.1.1 Pinhole Camera Model

Pinhole camera forms images similar to our eyes. Light rays from the real world objects pass through a small aperture and form an inverted image at the other side of the camera. The camera matrix maps the 3D world scene into image plane. This camera matrix is calculated by using intrinsic and extrinsic parameters. The real world points are transformed into camera coordinate by using extrinsic parameters and the camera coordinates are mapped into the image plane by using intrinsic parameters.

$$P = \begin{bmatrix} R \\ t \end{bmatrix} K \tag{1}$$

P is the camera matrix. Extrinsic parameters include rotation R, and translation t. K is intrinsic parameter. The intrinsic parameters include the five internal parameter which are focal length, the optical center and the skew coefficient.

$$K = \begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix} \tag{2}$$

The internal parameters are in pixel coordinates which are obtained by the affine transformation.

2.1.2 Lens Distortion

The real camera lens must include distortion coefficients since lenses are used in real camera model.

Radial Distortion

Light rays bend more at the edges of the lens than they do at the center. The smaller the lens, the greater the distortion.

$$x_{\text{distorted}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3}$$

$$y_{\text{distorted}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{4}$$

k_1, k_2, k_3 are radial distortion coefficients. Normally only two coefficients are sufficient. For wide angle lenses, larger distortion occurs and all three coefficients are needed. x and y are undistorted pixel locations.

Tangential Distortion

Tangential distortion occurs when the lens and image plane are not parallel.

$$x_{\text{distorted}} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \tag{5}$$

$$y_{\text{distorted}} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \tag{6}$$

p_1 and p_2 are tangential distortion coefficients of the lens.

where;

$$r^2 = x^2 + y^2 \tag{7}$$

2.1.2 Calibrating The Camera

In calibrating the project camera, 39 test images of 8x6 chessboard are used for better results. 2D image points can be found from the image. For 3D world points (object points), an array is created with points (0,0,0),(1,0,0),(2,0,0)…. Z is always zero since the chessboard was kept stationary at XY plane. To find patterns in chessboard, openCV functions are used to find the chessboard corners and draw the pattern.

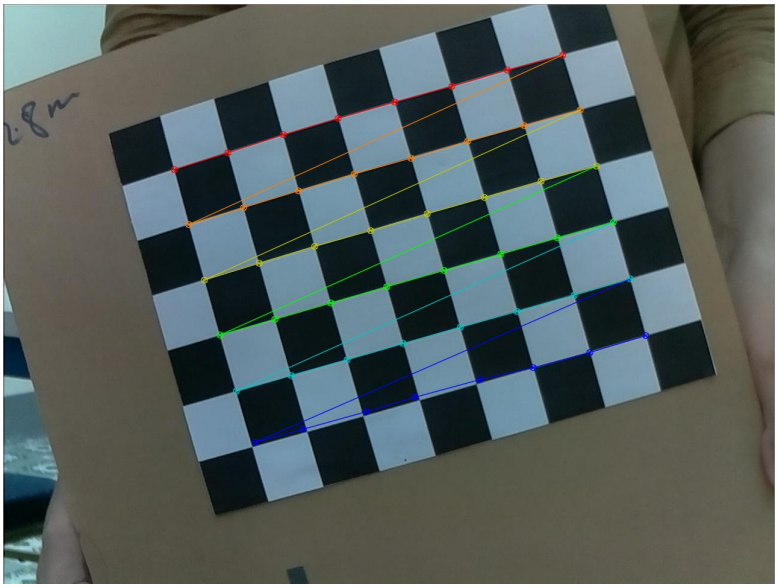


Figure (2.1) Chessboard Pattern

In calibration process, camera matrix, distortion coefficients, rotation and translation vectors which are resulted from the openCV camera calibrate function, are used to undistort the camera image.

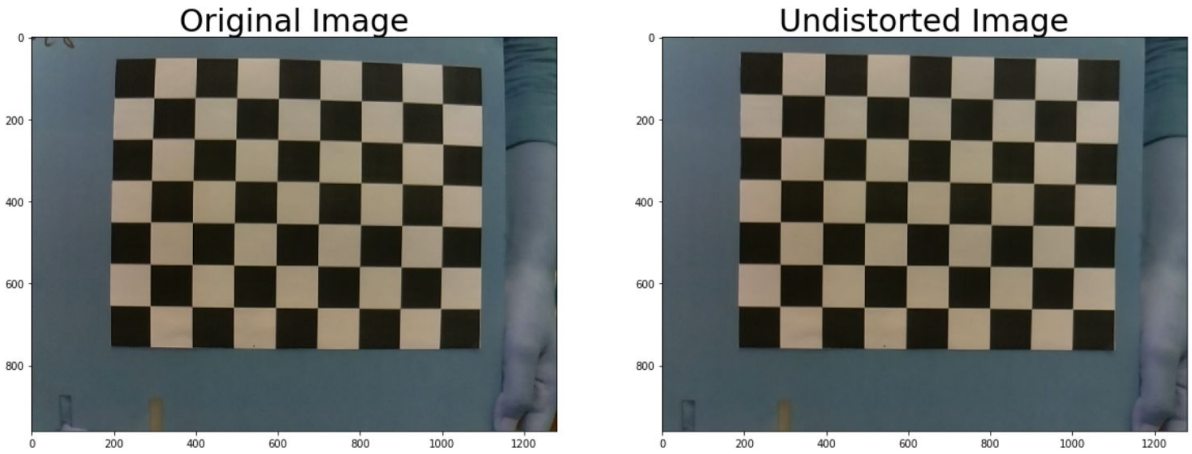


Figure (2.2) Undistorted Image

2.2 Image Processing Pipeline

The goal for the image processing process is to find out the cross-road-error, calculate the set point and then feed the data to the controller. In this report, a method of processing for only a small region of interest was proposed for faster analysis and better performance. For this project, OpenCV library was used for the software implementation. Figure 2.3 shows the flow of the whole image processing step.

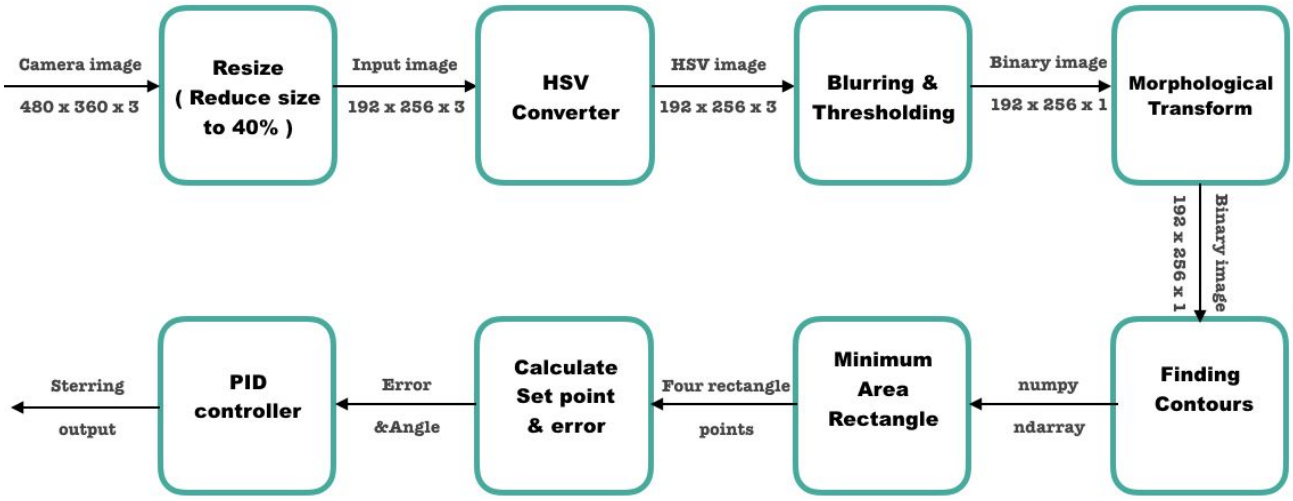


Fig (2.3) Flow diagram for the image processing step

2.2.1 Resizing Image

Better performance means faster calculation, low latency and higher accuracy. In the image processing, faster calculation can be obtained with lesser image resolution. Thus, resizing image to smaller dimension results in lesser calculation and thus faster performance. In this project, the original input image size has (480x640x3) pixel values but got resized to the (192x256x3) pixel values which means that 0.4 percentage reducing calculation time for the code to follow.

2.2.2 Image Thresholding

This is a challenging stage for the whole image processing. In the proposed environment, shadow, light and reflection from the lane line should be considered.

So, compared to thresholding in the ordinary RGB color space, better results can be obtained from thresholding the HLS or HSV color space because it is mostly invariant to the lighting condition of the lane line. For this project, HSV color space is chosen. Figure 2.4 represents the illustration of the HSV color space.

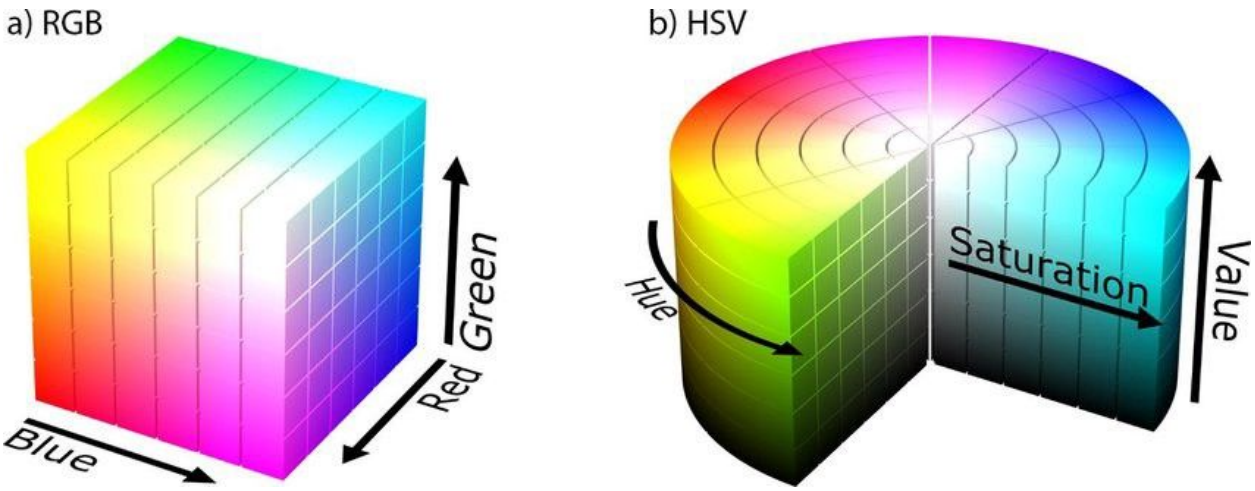


Figure (2.4) HSV Color Space

The conversion from RGB color space to HSV color space is given by

$$R' = R/255$$

$$G' = G/255$$

$$B' = B/255$$

$$Cmax = \max(R', G', B')$$

$$Cmin = \min(R', G', B')$$

$$\Delta = Cmax - Cmin$$

$$H = \begin{cases} 0^\circ & , \triangle = 0 \\ 60^\circ \times (\frac{G' - B'}{\triangle} mod 6) & , C_{max} = R' \\ 60^\circ \times (\frac{B' - R'}{\triangle} + 2) & , C_{max} = G' \\ 60^\circ \times (\frac{R' - G'}{\triangle} + 4) & , C_{max} = B' \end{cases}$$

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\triangle}{C_{max}} & , C_{max} \neq 0 \end{cases}$$

$$V = C_{max}$$

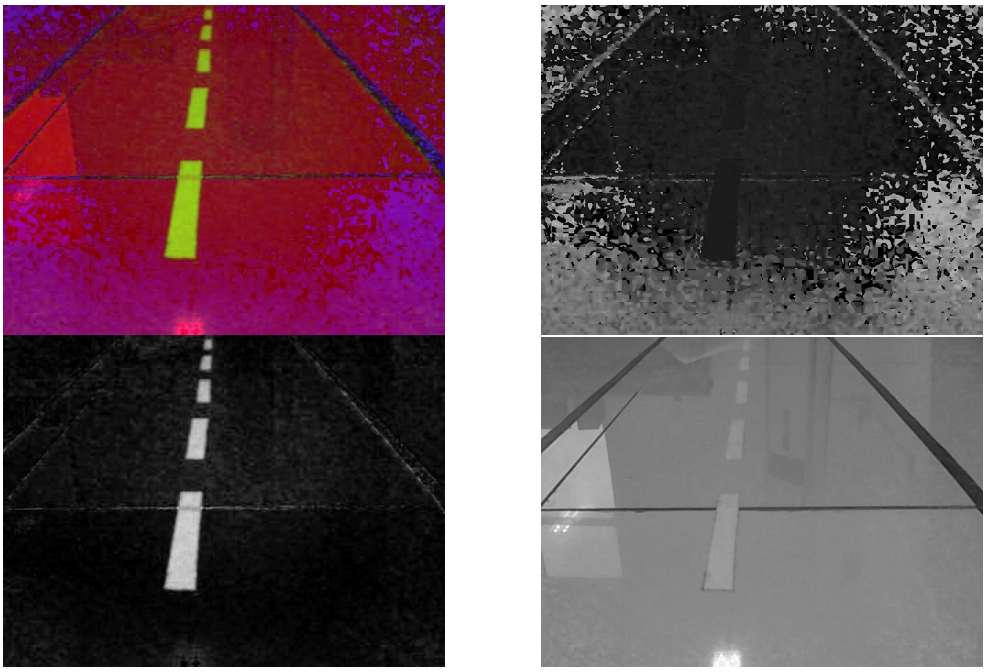


Figure (2.5) Image in HSV color space (2.6) Hue (2.7) Saturation (2.8) Value in HSV space

In figure 2.5, the original image is converted from the RGB color space to the HSV color space. In figure 2.6, an individual *Hue* space is extracted from the HSV color space and displayed in the gray scale form. Figure 2.7 and 2.8 are in *Saturation* and *Value* spaces from the HSV color space. The images shown that saturation space can be used for better lane detection. And the result can get even better with the preprocess step like blurring the image. Generally used blurred algorithm is the Gaussian blurred and was used for this purpose.

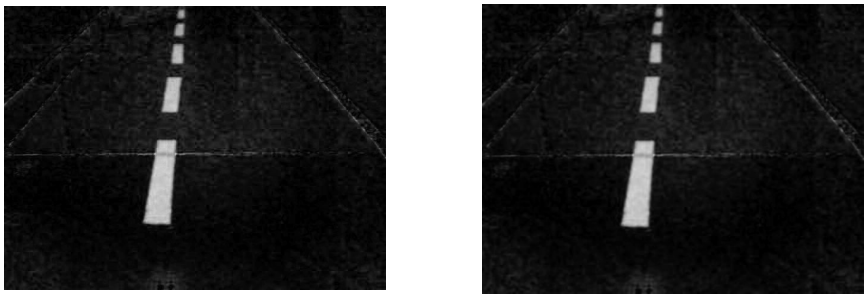


Figure (2.9) Image without blurred (2.10) Image with blurred

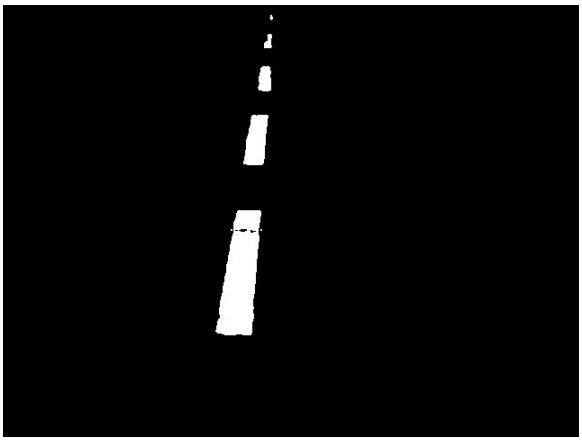


Figure (2.11) Thresholded image

The above figures are the image before thresholded and after thresholded. Usually, the difference in blurring can be seen by the amount of noise reduced after thresholding. Figure 2.11 shows the result of the thresholded image.

2.2.3 Morphological Transformations

After thresholding, a binary image is obtained. Morphological transformation is then performed to further reduce noise and improve outcome of the image.

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is the original image, second one is called ***structuring element*** or ***kernel*** which decides the nature of operation. Two basic morphological operators are *Erosion* and *Dilation*. In both Erosion and Dilation, the resulted image will be obtained from the kernel sliding through (convolution) the original image, the only difference is that in erosion, the pixel will be considered as one only if all the pixel under the mask is one. Otherwise, it will be set to zero. Thus, it is often used to erode away the boundaries of the foreground objects and remove noise. The erosion equation is given by

$$A \ominus B = \{z \mid (B)_z \subseteq A\} \tag{8}$$

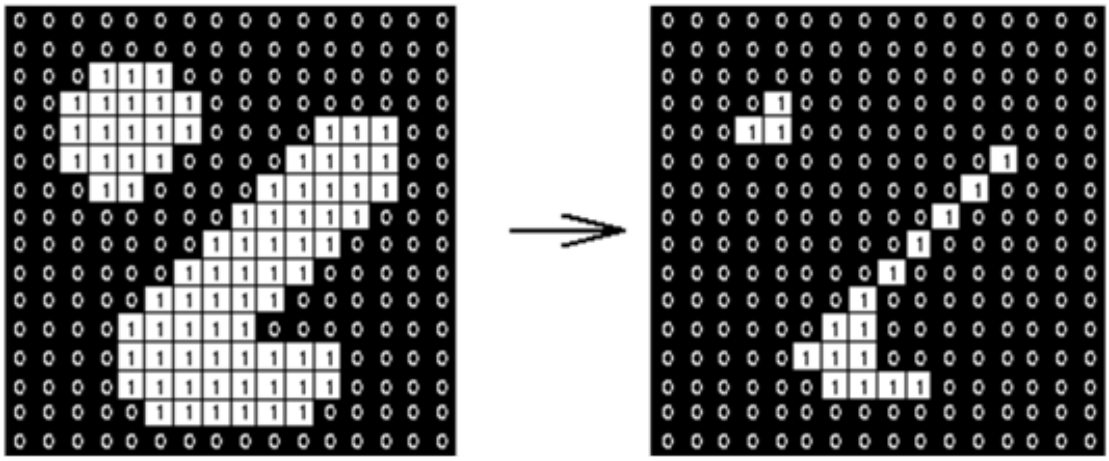


Figure (2.12) Effect of Erosion with 3x3 square structural element

After Erosion, Dilation often come. Erosion removes the noise from the image. But it also reduce the lane width as it goes. Dilation can be used to increase the lane pixel width. A pixel will be set to one if at least there is one pixel under the mask that is one in the Dilation process. So, it results in increasing the white region in the image or size of the foreground.

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\} \tag{9}$$

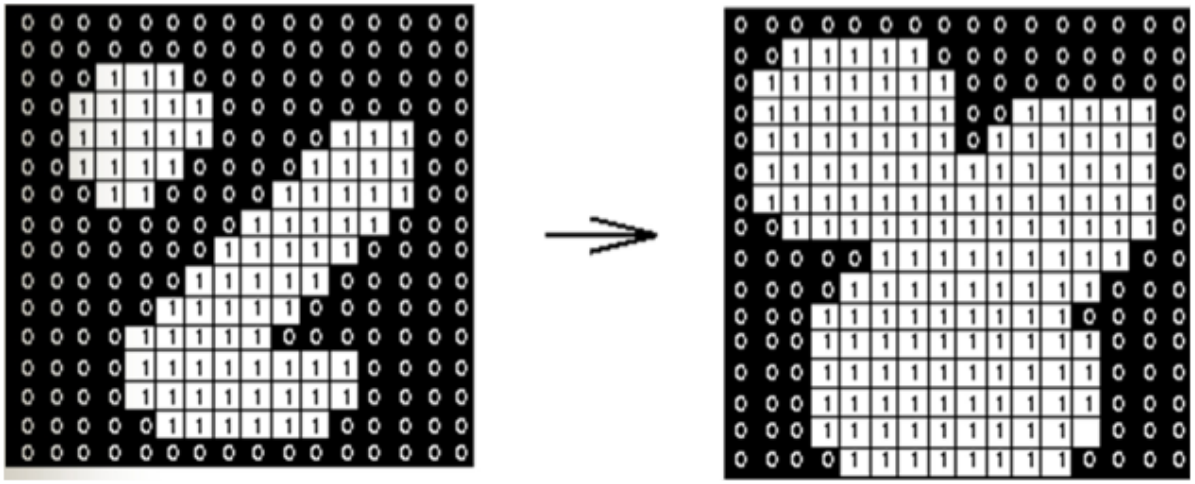


Figure (2.13) Effect of dilation with 3x3 square structural element

Below is the application of the morphological to the project lane line.

Figure (2.14) Original image (2.15) Dilated image

2.2.4 Contours point and set point

The very first objective of this project in computer vision is to detect lane lines pixel in the image frame and outputs the set point needed for lane travel. Contours concepts are useful here to detect the lane boundary points. Contours can be simply explained as the curve joining all the continuous points having the same intensity. Often used to determine the boundary of the object used for segmentation and other purposes. But contours usually find all the boundary point which can slow down the code continuous on. So Minimum Area Rectangle function is being used. The output is the rectangle provided with the angle of the rotation of the rectangle formed by the first encounter pixel block. Figure 2.16 represents the thresholded image with all the contours points highlighted. Figure 2.17 shows only the first block rectangle which center is calculated as the set point.

Figure (2.16) With all Contours points (2.17) with only minimum Area Rectangle

2.2.5 Experiments and Results

To achieve self driving with computer vision using only one camera can be proven difficult in accuracy and efficiency. The first one in that it is hard to detect the depth of the lane pixel found with just one camera, the second in that the reflection of the lane, the disturbance found in the way of driving. With various color space and parameters tuning, decent results can be found for this project. But it still needed a lot of time with other noises added to the frames.

3. Control System and Kinematics Modeling

Image processing pipeline outputs the cross track error which is the error value of the robot that is staying in the middle of the lane or not. PD Controller is used to minimize this cross track error and outputs the appropriate rotational velocities of the robot. Robot's rotational velocities are then converted to each wheel's angular velocities by using mobile robot kinematics model.

3.1 PID Controller

With its three-term functionality covering treatment to both transient and steady-state responses, proportional-integral-derivative (PID) control offers the simplest and yet most efficient solution to many real-world control problems. A PID controller may be considered an extreme form of a phase lead-lag compensator with one pole at the origin and the other at infinity. Similarly, its cousins, the PI and the PD controllers, can also be regarded as extreme forms of phase-lag and phase-lead compensators, respectively. A standard PID controller is also known as the "three-term" controller, whose transfer function is generally written in the form given by (1)

$$G(s) = K_p + K_i\frac{1}{s} + K_d s \tag{10}$$

where ***Kp*** is the proportional gain, ***Ki*** the integral gain and ***Kd*** the derivative gain. The project uses PD controller for the robot motor control and its architecture can be simplified as shown in Figure 3.1. The controller minimizes the error between

the desired setpoint value and the actual robot's positions and outputs the rotational velocities to inform the robot where to move.

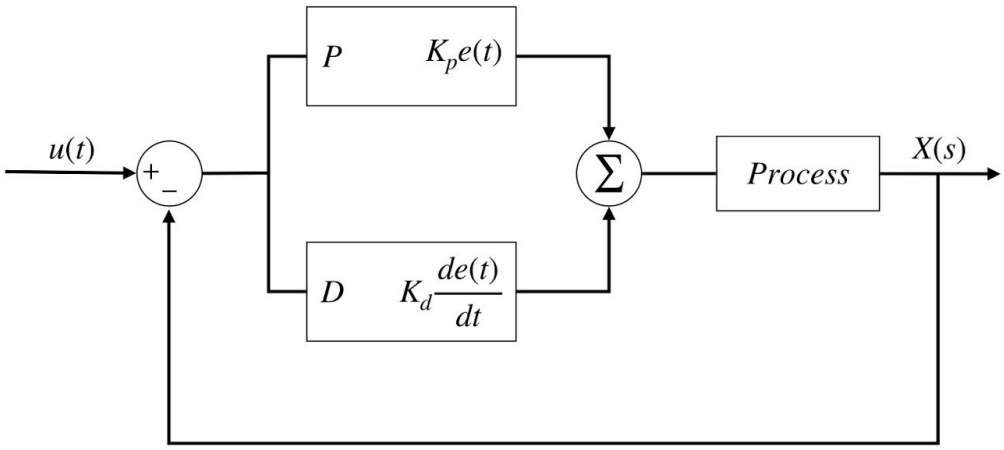


Figure (3.1) Control architecture of the motor controller

3.2 Kinematic Model of the mobile robot

Kinematic modeling is the study of the motion of mechanical systems without considering the forces that affect the motion. The main purpose of kinematic modeling is to represent the robot velocities as a function of the driving wheels velocities along with the geometric parameters of the robot. Figure 3.2 represents the kinematic model of a differential-drive robot.

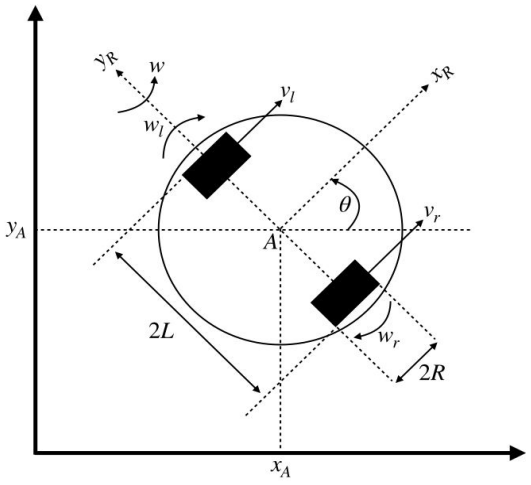


Figure (3.2) Kinematic model of a differential-drive robot

The linear velocity of the differential drive mobile robot in the Robot Frame is, therefore the average of the linear velocities of each driving two wheels

$$v = \frac{v_r + v_l}{2} = R\frac{w_r + w_l}{2} \tag{11}$$

and the angular velocity of the differential drive mobile robot is

$$w = \frac{v_r - v_l}{2L} = R\frac{w_r - w_l}{2} \tag{12}$$

Rotational velocities, resulted from the PD Controller are then converted to each wheel angular velocity by using above kinematic equations.

4. Robot Operating System (ROS)

Writing software for robots is difficult, particularly as the scale and scope of robotics continue to grow. Different types of robots can have widely varying hardware, making code reuse nontrivial. To meet these challenges, many robotics researchers, have previously created a wide variety of frameworks to manage complexity and facilitate rapid prototyping of software for experiments, resulting in the many robotic software systems currently used in academia and industry. Each of these frameworks was designed for a particular purpose, perhaps in response to perceived weaknesses of other available frameworks, or to place emphasis on aspects which were seen as most important in the design process. ROS, the framework described in this paper, is also the product of tradeoffs and prioritizations made during its design cycle.

4.1 ROS Overview

ROS is not an actual operating system, but rather a meta-operating system. Simply put, ROS works on top of other operating system and allows different processes to communicate with each other during runtime. ROS is also a set of tools that provides the functionality and services of an operating system on a single or multiple computers. These services include abstraction of the hardware, exchange of messages between processes, management of packages, etc.

The philosophical goals of ROS can be summarized as:

Peer-to-peer. ROS nodes are units of execution that communicate with each other directly or via publish/subscribe mechanisms.

Tools-based. It uses a microkernel design and several small tools and modules.

Multi-lingual. It has support for C++, Python, Octave, LISP and other languages.

Thin. It uses a catkin build system to provide code segmentation in terms of packages and libraries.

Free and open source. ROS is publicly available under a BSD license.

The fundamental concepts of ROS implementation are *nodes*, *messages*, *topics* and *services*. Figure 4.1 illustrates the role of ROS master node: *roscore*, which is an essential part of each ROS-based program. The roscore master node must be running for ROS nodes to communicate. When roscore is active, nodes can exchange messages by publishing and subscribing to certain topics or by directly invoking the services and actions of the other nodes. Figure 4.2 illustrates the ROS concepts (nodes, messages and topics) and how they correlate.

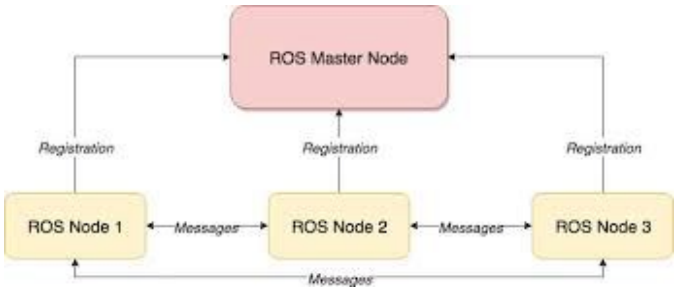


Figure (4.1) Illustration of ROS nodes and messages

Figure (4.2) Visualisation of ROS concepts

4.2 ROS Development

In order to implement the lane lines following process, robot operating system (ROS) is used as a middleware of the system. System architecture of the following lane line process is shown in figure 4.3.

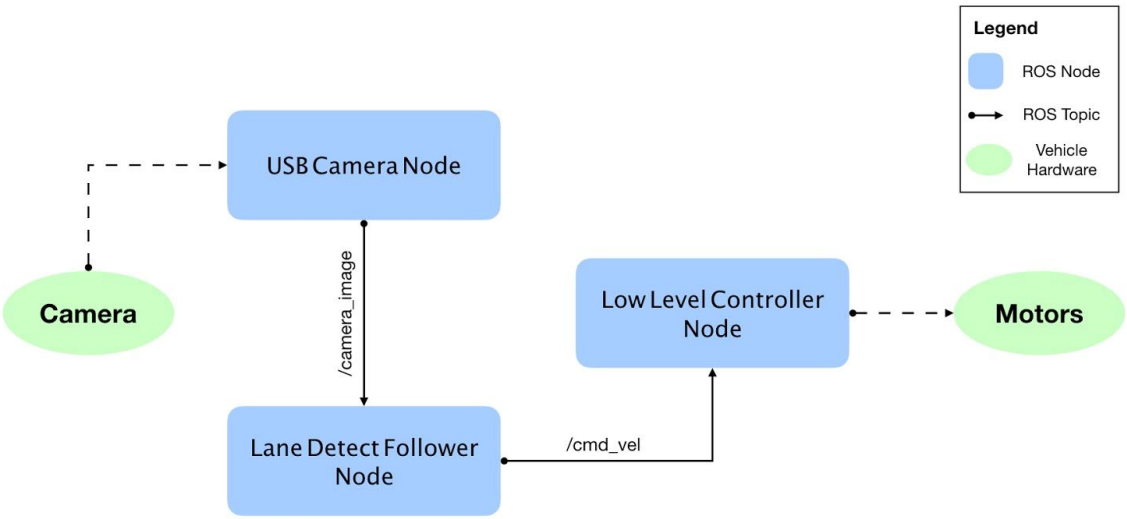


Figure (4.3) System Architecture Diagram

Three ROS nodes are connected with each other to develop the process. They are summarized as follows:

I. USB Camera Node

USB Camera Node interfaces with the standard USB cameras and publishes images as sensor_msgs::Image topic.

II. Lane Detect Follower Node

Main lane following node for the project. This node subscribes the images topic and publishes the robot rotational velocities.

III. Low Level Controller Node

This node subscribes the robot rotational velocities and calculates the mobile robot kinematics in order to find the wheels' rotational velocities. It then interfaces with the motors to achieve the desired wheels' rotational velocities.

Figure 4.4 illustrates the rqt_graph for the computation of different ROS nodes and topics.

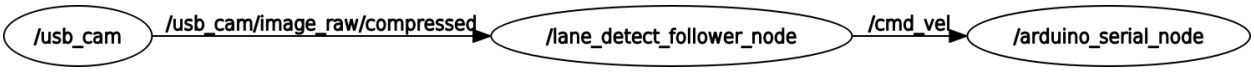


Figure (4.4) ROS Computation Graph

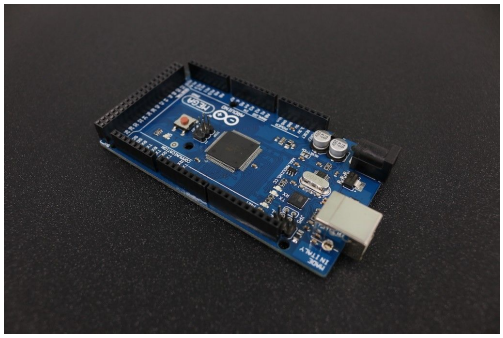
5. Development of Embedded System and Robotic Mechanism

5.1 Components of embedded system

One of the important tasks for executing a project is to choose the most suitable components. In other words, the chosen components should have specifications that can well-perform the project tasks. Moreover, they must be energy cost-effective too. The components in this project meet both requirements.

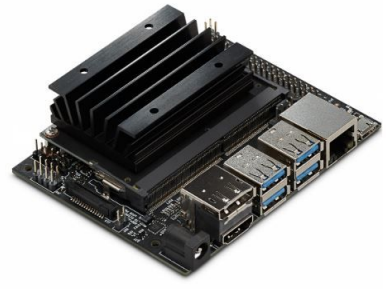
I. Arduino mega board

The arduino mega is an open-source board.The mega board is preferred in this project because it has more memory space and I/O pins as compared to other boards available.



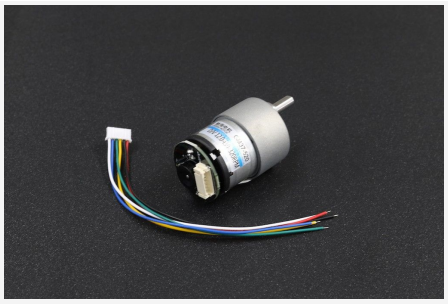
II. Nvidia Jetson Nano

Jetson nano is preferred in this project because it is a high-performance microprocessor with the ability to carry out modern AI algorithms fast and processes several high resolutions sensores simultaneously. And it consumes less power (5 to 10W).



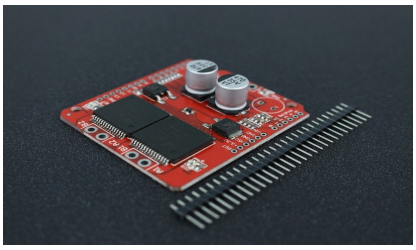
III. DC motors

GM37-520 geared DC motor is used since it can provide sufficient speed and high torque required by the robot since the robot has considerable amount of inertia and should have motion neither too fast nor too slow. Moreover, they are very small, lightweight and highly efficient.



IV. VNH2SP30 Monster motor driver

The *VNH2SP30* is a full bridge motor driver intended for a wide range of automotive applications.The input signals INA and INB can directly interface to the microcontroller to select the motor direction and the break condition. This driver is suitable in this project using two motors because it can withstand high current draw which is around 30A for dual motor drive.



V. Li-Po battery

The Li-Po battery 11.1V, 25C, 2200mah is chosen in this project since the recommended voltage for the motors is 12V and the battery can provide a maximum current of around 50A(25C × 2.2ah=55A) surpassing the maximum current draw of the two motors.

VI. Camera

Logitech webcam c170 is used in the project for vision. It has a resolution of 640✕480, 5MP.

The block diagram of the robot embedded system is shown below.

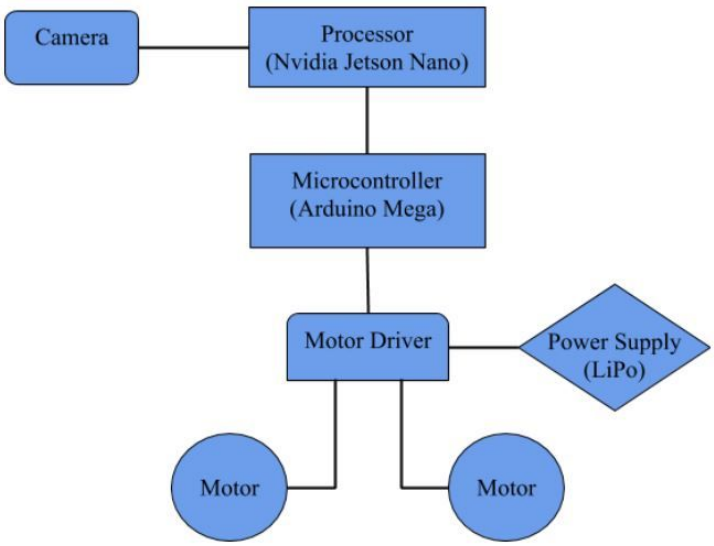


Fig (5.1) Simple block diagram of embedded system

5.2 Robotic Design and Mechanism

The design of the robot is based on the objectives of the project. In addition, the energy and cost efficiency are also taken into consideration. The mobility of the robot is achieved by using differential drive system. The main advantage of this system is that it requires no steering components and actuators, and hence, reduces the weight of the robot considerably. In the project, the differential drive system with two motors is used to reduce the complexity of the control system geometry. Moreover, turning and rotating are also carried out just by driving the two motors at different speeds or directions. Figure shows the simple mechanism of the differential drive system. The most common disadvantage of the dual motors-differential driver system is the difficulty of moving straight. This problem is overcome by using PID controllers and tuning the speed of the motors with encoders.

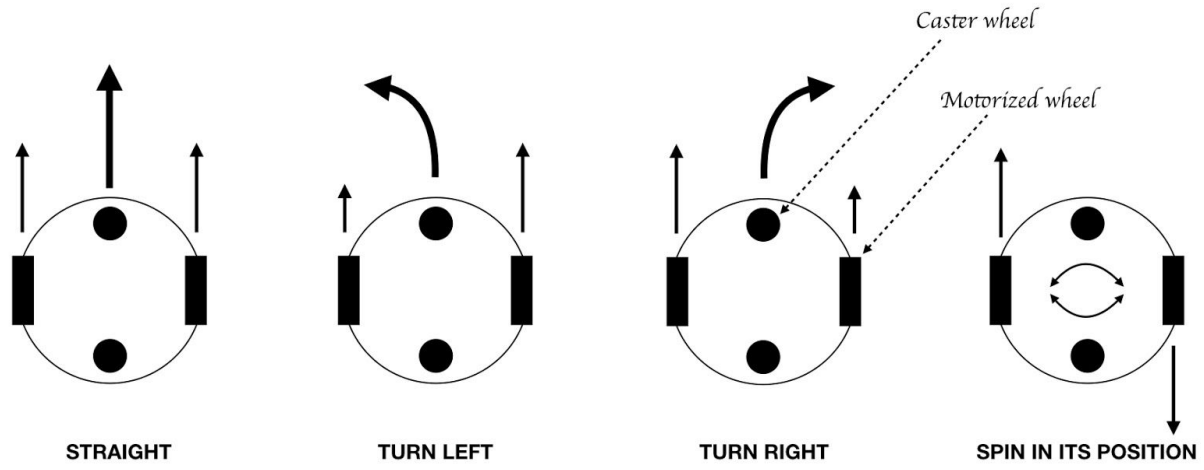


Figure (5.1) Mechanism of dual differential drive system

The two wheels are directly attached to the motor axis pole. Both axes of the wheels are parallel and concentric and pass through the center of the robot as shown in figure 5.1. This gives high manoeuvre to the robot. The balance of the robot is attained by two caster wheels,mounted in the shape that the line connected between the two caster wheels is perpendicular to the axis of the motorized wheels.

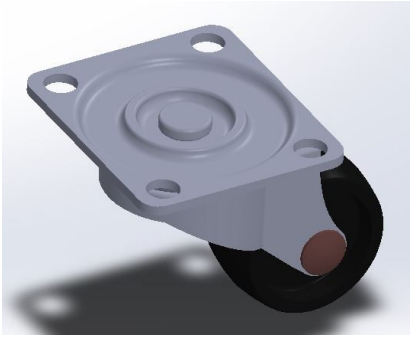


Figure (5.2) Caster wheel

The main frame of the robot is built with acrylic plates in the purpose of lighter weight, greater strength and hardness. Since several electrical components are to be used, space is required in the robot. Three circular acrylic plates are connected layer by layer with spacers between them. This, not only gives enough space for the embedded system, but also reduces the horizontal length and width of the robot. The overall structural body is in the form of a cylinder.

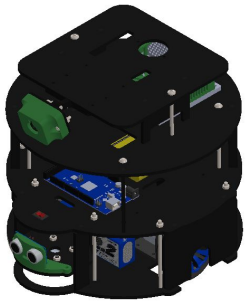


Fig (5.3) Isometric view (5.4) Front view of the robot in a simulator

The camera is mounted on the front side of the robot, at a level high enough so that the camera can capture most suitable images and the camera calibration is carried out efficiently. The height of the camera also plays an important role in this project. If the camera is too high, the real-time images captured would be too far, and too near if the camera is too low.

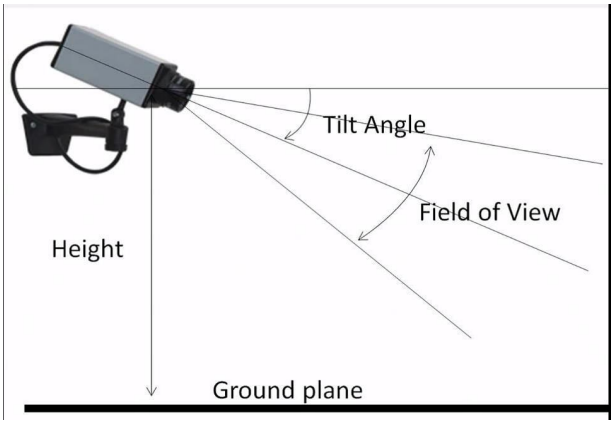
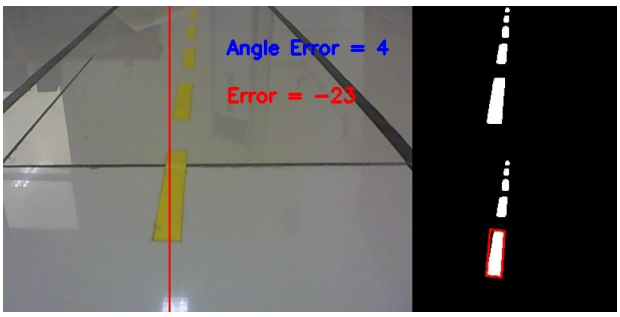
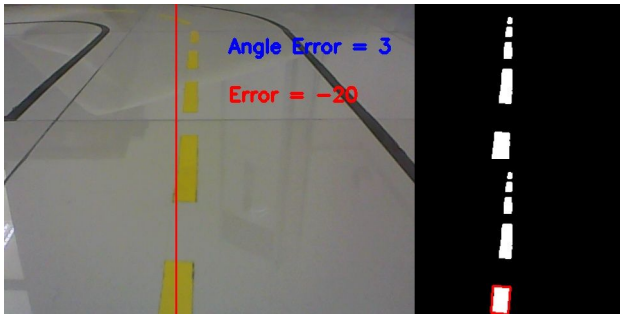


Fig (5.5) Height of a camera and field of view

6. Experiments and Results

A clear and precise result can only be obtained with the appropriate hardware, low latency algorithm and tuned parameters. Selection of the right hardware comes from the experience looking from the specification of the product. While in coding algorithms and tuning parameters, it takes repeated experiments for the best result. In image processing step, right threshold value, right kernel structure and filter, order of operations and methods used for each operation and many other more can all be tuned. There are also trades off between low latency requirement and better result. After many experiments and results testing, the algorithm best suited for this project give out the following outputs.





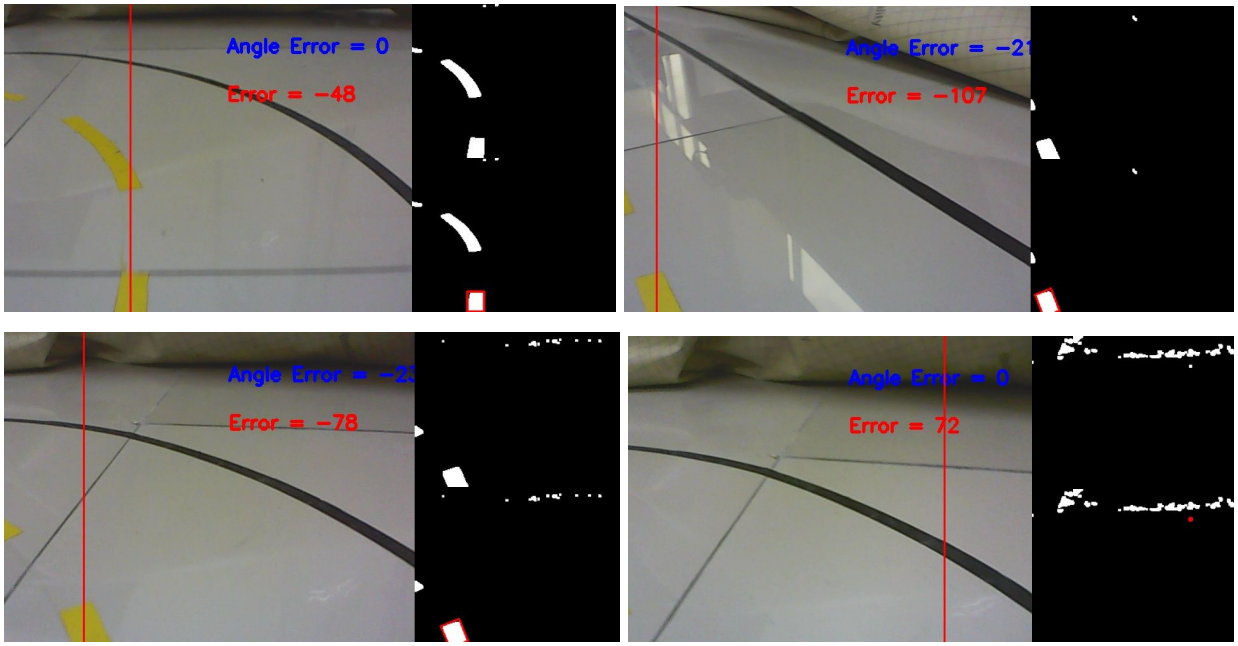
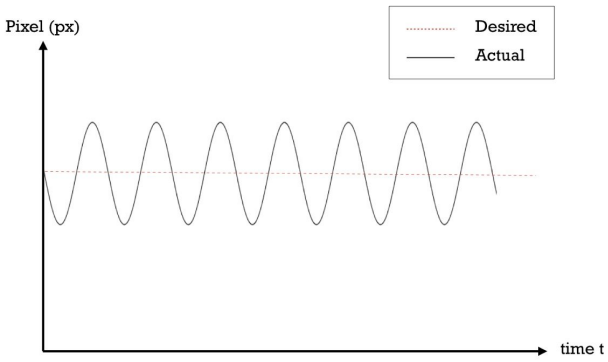


Fig (6.1) (6.2) (6.3) final result image with straight lane (6.4)(6.5)(6.6) final result image in curve

The result for the project is pretty decent in the lighting varying condition from the above conditions shown. As such, the image processing step is well performed. And then it all comes down to the control technique and the values it used. PD controller was employed for this project and the respective controller gains were adjusted to obtain the desired speed with curve able movement as shown in Figure 6.7. And the result is the smooth motion in both straight and curved lane lines.



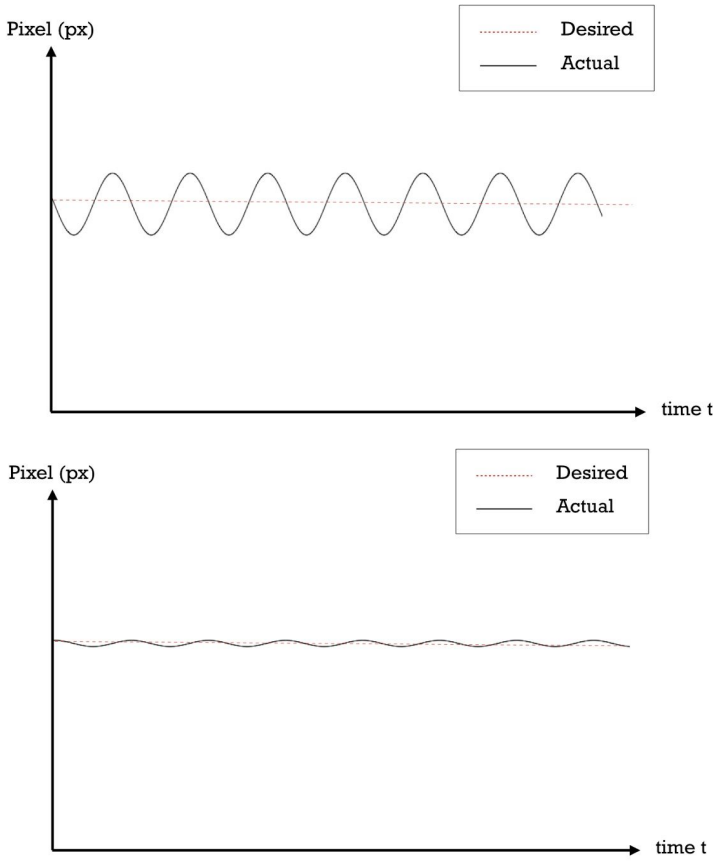


Figure (6.7) Gains Kp = 0.1, Kd = 0.0 (6.8) Gains Kp = 0.05, Kd = 0.05 (6.9) Gains Kp = 0.01, Kd = 0.05

7. Conclusion and Future Works

This project focuses on the perfect combination of computer vision, PID control and Robot Operating System (ROS) including some Mechatronic subjects added by some new methods and techniques. To be exact, an autonomous differential-drive mobile robot is built in order to perform lane line detection using various image processing techniques and a communication framework called ROS. Since autonomous vehicles and robots become new technological development, they are being challenged to do more tasks such as classification of different types of objects detected on the road, avoiding obstacles, obeying transportation rules, traffic light rules and so on. In this case, further developments of current project

will be done by analysing autonomous driving problems and finding solutions in order to perform various tasks in the future.

Acknowledgements

We would first like to thank our advisors and Mechatronics teachers at Yangon Technological University. They steered us in the right direction whenever we had some questions about our project. Their guidance helped us out all the time and we could not have imagined having better advisors for our project.

Our sincere thanks also goes to our friends and classmates especially Htet Arkar for his assistance with the robot designing process. Without their precious support, it would not be possible to complete this project.

References

[1] **Lane Detection for Autonomous Vehicles using open CV Library** by Aditya Singh Rathore (B.tech,J.K. Lakshmipat University)

[2] Master Universitario en Ingenieria del Software- European Master in Software Engineering Thesis Title: **Master development of autonomous Driving using Robot Operating System**(Thesis no: EMSE-2018-3)

[3] **Self-Driving and Driver Relaxing Vehicle** by Qudsia Memon, Shahzeb Ali, Wajiha Shah, Muzamil Ahmed and Azam Rafique Memon(all are from the department of Electronic Engineering)

[4] **AutonomousCar: Past, Present and Future : A review of the Developments in the last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology** by Keshav Bimbraw (Mechanical Engineering department, Thapar University, P.O. Box 32, Patiala, Punjab, India)

[5] **Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art** by Joel Janaia, Fatma Guney, Aseem Behla, Andreas Geiger (Autonomous Vision Group, Max Planck Institute for Intelligent Systems,

Spemannstr. 41, D-72076 Tubingen, Germany bComputer Vision and Geometry Group, ETH Z urich, Universitatstrasse 6, CH-8092 Zurich, Switzerland)

[6] **Autonomous Vehicle And Real Time Road Lanes Detection And Tracking** by Farid Bounini, Denis Gingras(University de Sherbrooke, Canada) and Vincent Lapointe, Herve Pollart(Opal-RT Technologies Inc, Montreal, Canada)

[7] **Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework** by Rached Dhaouadi and Ahmad Abu Hatab (College of Engineering, American University of Sharjah, Sharjah, UAE)

[8] **Nonlinear Motion Control of Mobile Robot Dynamic Model** by Jasmin Velagic, Bakir Lacevic and Nedim Osmic (University of Sarajevo Bosnia and Herzegovina)

[9] **Design of an Adaptive PD Controller for the Weight-Independent Motion Control of a Mobile Robot** by Hwan-Joo Kwak and Gwi-Tae Park (Department of Electrical Engineering, Korea University, Republic of Korea)

[10] **Udacity Self-Driving Car Engineer NanoDegree** (www.udacity.com)

[11] **OutOfTheBots** (YouTube Channel)

****