DRISHTI MINI PROJECT

Autonomous Car

DRISHTI

A Revolutionary Concept

SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY • SURAT •

Contents

1 TEAM MEMBERS

1.1 Mentors

DRUV OZA
HARSH AGRARWAL
RAJ SINHA

1.2 Students

AMIT YADAV
EESHAN BHARGAVA
HARDIK JAIN
JAY WALA
SHIKHA KUMARI BHARATI
SUYOG POKALE
SWARA JAIN
TEENA DHOKE

2 Introduction

An autonomous car is a vehicle capable of sensing its environment and operating without human involvement. A human passenger is not required to take control of the vehicle at any time, nor is a human passenger required to be present in the vehicle at all. An autonomous car can go anywhere a traditional car goes and do everything that an experienced human driver does.

3 Abstract

The aim of this project is to track a given path by an autonomous car. This project explored the simulation of autonomous car in software called ROS. The algorithm need to be designed in such a way that it doesn't leave it's path and take turns or varying speed according to path. The problem statement track contains an overhead bridge and a tunnel. This also needs to be considered as well and make progress in algorithm in such a way so that it will make balance on the track.

software used for this project is ROS which is open source robot simulation software.

4 Software

4.1 Ubuntu

Ubuntu is an open source Debian-based Linux distribution.Ubuntu is recommended as it supports ROS and Gazebo over windows.

4.1.1 Installation Ubuntu 20.04 LTS

Process: DUAL BOOTED
Software download Link: `https://ubuntu.com`
BIOS:(UEFI)
YouTube link to be followed How to Dual Boot Ubuntu 20.04 LTS and Windows 10 — A Step by Step Tutorial — [2021] - UEFI Linux - YouTube

- After downloading the Ubuntu 20.4LTS ISO image, make a bootable USB drive by formatting it using software and burn the ISO image to the USB drive

- Make partitions and allocate free space for ubuntu in drive. and restart the pc and open the boot menu then select the bootable drive.
- Select normal installation and install third party software. then continue.
- Choose ‘something else’ and choose the free space partition made initially. Select ext4 journaling file then mount point as “ /” then continue.
- Begin installation
- Select your time zone
- Login details
- Installation complete
- Restart the PC, remove USB drive,GRUB will appear then select Ubuntu

4.2 ROS

The Robot Operating System (ROS) is an open-source framework that helps researchers and developers build and reuse code between robotics applications. controller can control an autonomous car or bot in ROS whose motion is controlled by a program which is made by c++, python etc. We are using ROS2 for simulating our car in a given path. The environment of ROS software ‘Gazebo’ in which controller can see how our car is running on path is shown in figure.With Gazebo you are able to create a 3D scenario on your computer with robots, obstacles and many other objects. Gazebo also uses a physical engine for illumination, gravity, inertia, etc. You can evaluate and test your robot in difficult or dangerous scenarios without any harm to your robot.

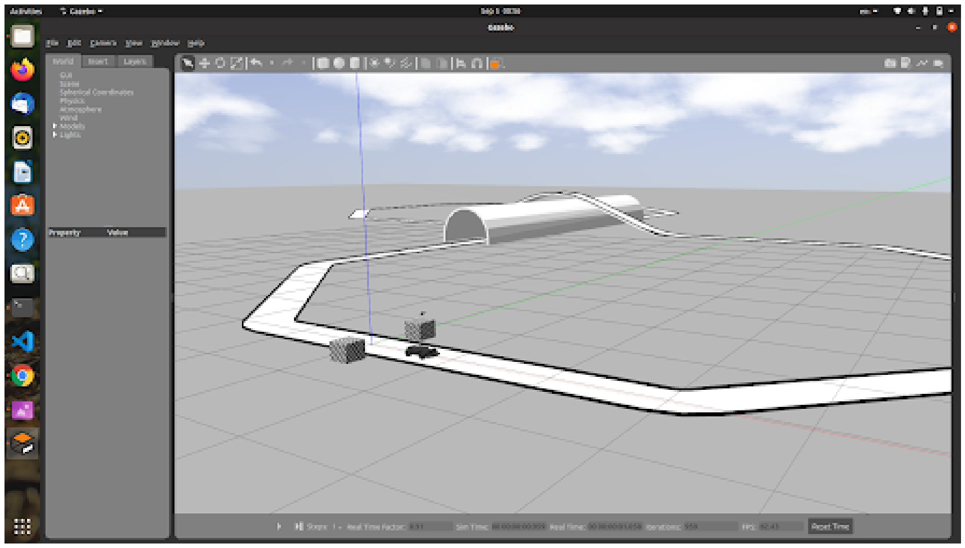
Figure 1: Track

4.2.1 Installation

Reference link:
https://docs.ros.org/en/crystal/Installation/Linux-Install-Binary.html
Generation of SSH key:
In order to installation of ROS ssh key should be generated by run the following commands in terminal:

- ssh-keygen
- sudo apt install xclip git
- xclip -sel clip < /.ssh/id_rsa.pub

Adding SSH key to github:
For adding ssh key in github account copy contens of id_rsa.pub in clipboard so paste it in github. In order to copy content to clipboard installing xclip by following command:
$ sudo apt install xclip git
After installing copy contents of id_rsa.pub file on github by running following command on terminal:
$ xclip -sel clip < /.ssh/id_rsa.pub

Installation of ROS:

File required for installation: https://firebasestorage.googleapis.com/v0/b/gitbook-28427.appspot.com/o/assets%2F-MWeiLnwrIKZJWLFsXhf%2F-MXcfaeFblmznqth_0fk%2F-MXcgSRrBPzq5-K6O-C5%2Ffoxy_install_aim.sh?alt=media&token=2adf3a55-8463-4ff1-890e-67b6d32fe747

Commands used in terminal for installing ROS:

- cd /Downloads
- chmod a+x foxy_install_aim.sh
- ./foxy_install_aim.sh
- source ~/.bashrc

Installation of Gazebo:
Following commands are run in terminal in order to install Gazebo:

- cd ~
- mkdir -p robocon_ws/src cd robocon_ws/src
- git clone https://github.com/harshag37/sim_gazebo_bringup.git
- cd ~/robocon_ws
- colcon build –packages-select sim_gazebo_bringup –symlink-install
- echo ”source /home/$USER/robocon_ws/install/setup.bash” » ~/.bashrc
- source ~/.bashrc
- ros2 launch sim_gazebo_bringup sim_gazebo.launch.py (For lunching ROS)

4.2.2 Packages

A package can be considered a container for your ROS 2 code. If you want to be able to install your code or share it with others, then you'll need it organized in a package. With packages, you can release your ROS 2 work and allow others to build and use it easily.
Package creation in ROS 2 uses ament as its build system and colcon as its build tool. You can create a package using either CMake or Python, which are officially supported, though other build types do exist

4.2.3 Nodes

Each node in ROS should be responsible for a single module purpose (e.g. one node for controlling wheel motors, one node for controlling a laser range-finder, etc). Each node can send and receive data to other nodes via topics, services, actions, or parameters. A full robotic system consists of many nodes working in concert. In ROS 2, a single executable (C++ program, Python program, etc.) can contain one or more nodes.
A node may publish data to any number of topics and simultaneously have subscriptions to any number of topics.

4.2.4 Topics

Topics are one of the main ways in which data is moved between nodes and therefore between different parts of the system.

4.2.5 Publisher and Subscriber

Publisher gives data and Subscriber receives it. The same analogy can be seen in sensors(Publisher) exchange data to micro controllers(subscriber).

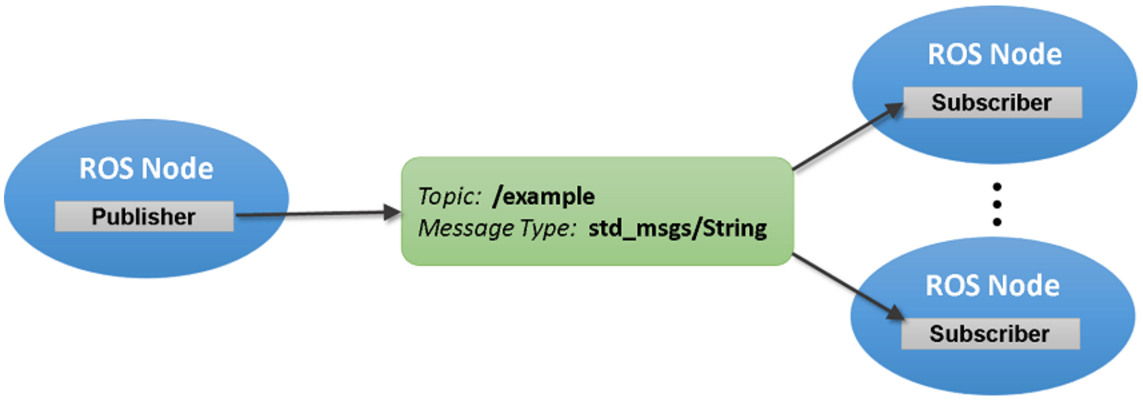

Figure 2: Caption

5 Concepts

Perception: Autonomous vehicle perception sensing involves the collection of data from vehicle sensors and the processing of this data into an understanding of the world around the vehicle. Autonomous cars have video cameras and sensors in order to see and interpret the objects in the road just like human drivers do with their eyes.

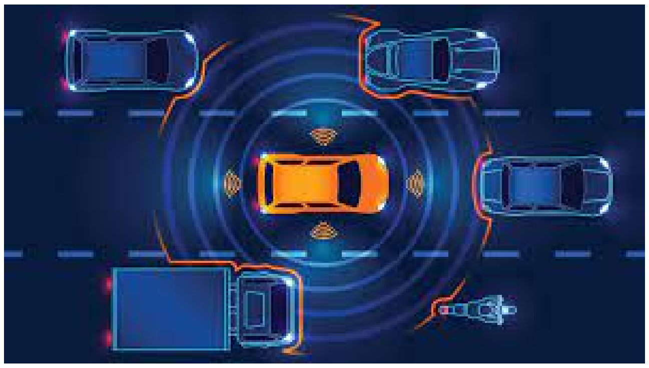

Figure 3: Perception

Localisation: Localization is the implementation of algorithms to estimate where is position of vehicle. Localization is a step implemented in the majority of robots and vehicles to locate with a really small margin of error.

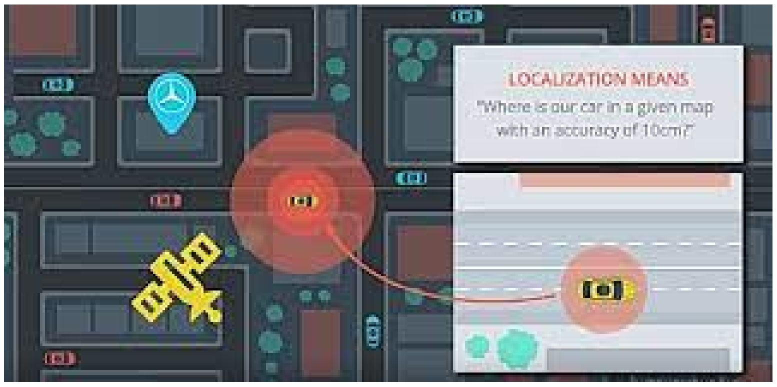

Figure 4: Localisation

Path planning: If a car doesn't know the path which it will track then controller have to do path planning. Path planning technology searches for and detects the space and corridors in which a vehicle can drive. It gives a feasible collision-free path for going from one place to another.

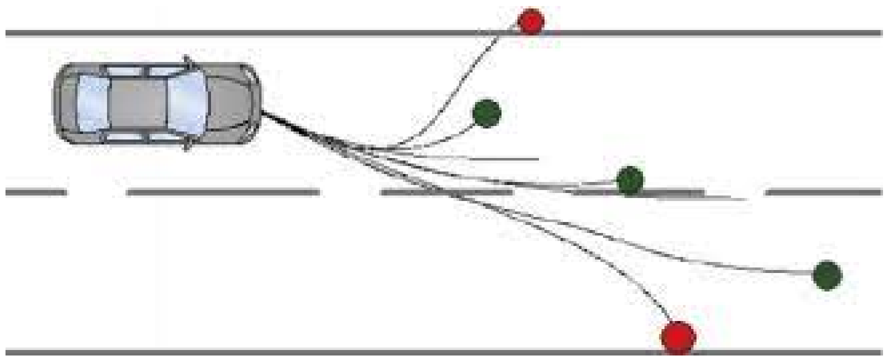

Figure 5: Path Planing

Motion Planning: controller can see that in any path there are turns, some sharp or soft, tunnels, bridges etc. When a car turns at any path we have to decrease its speed for turning smoothly and also control speed at bridges. This all things can be done by motion planning.

6 Gazebo Enviroment

This image describes the camera view of the car. The total path a car can see is shown in a green space which is marked by the edged coordinates of the shown track. The basic idea to move a car is to calculate the midpoint of the marked coordinates and ensure that the car move on the centre line

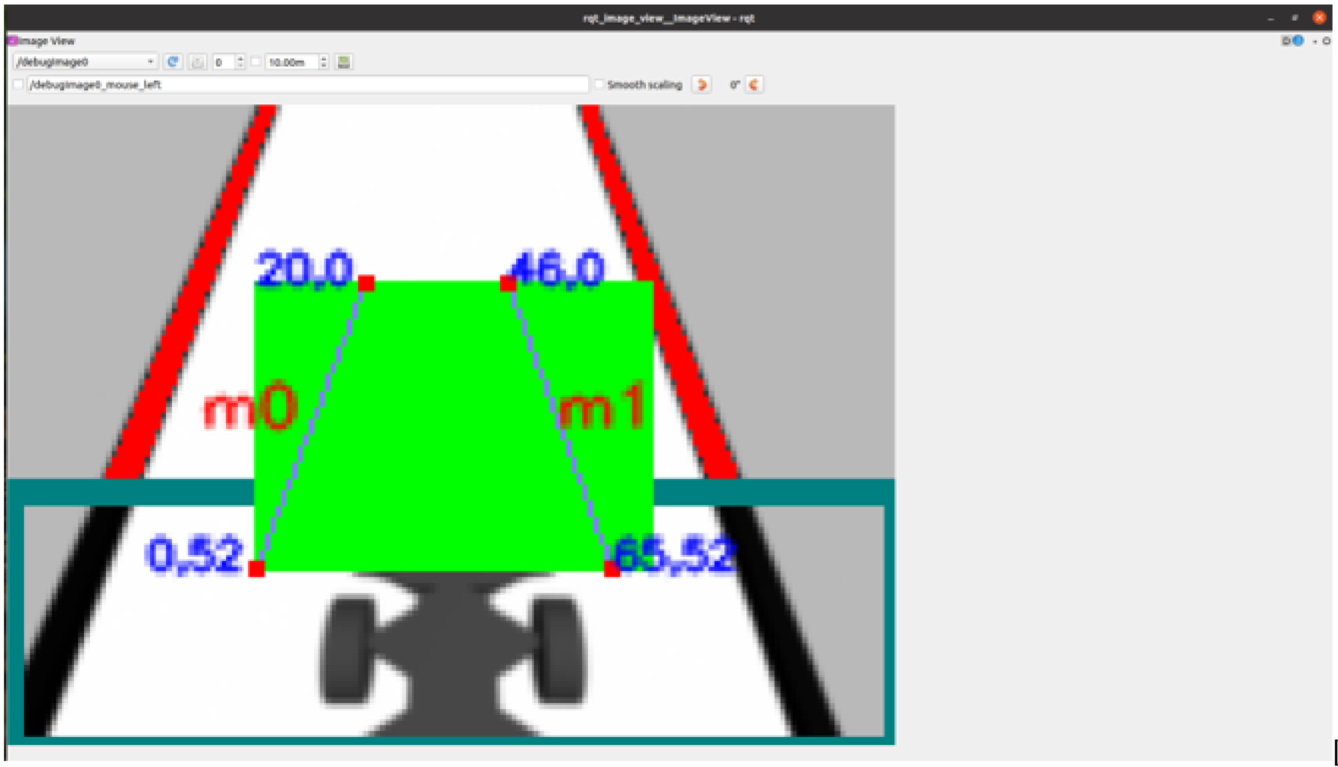

Figure 6: Vehicle With Co-ordinates

Frame width = width of the green window
Frame height = height of the green window
Window center = center of the frame width
(Implies that symbol) Window center = (frame width/2)

There are two vectors named m0 and m1 . The above mentioned coordinates are (m0x1, m0y1) and (m1x1, m1y1) and below mentioned coordinates are(m0x0, m0y1) and (m1x0 ,m1y0) . By calculating the mid point of both above and below coordinates their come a centre vector which is treated as a constant vector for the motion of car.

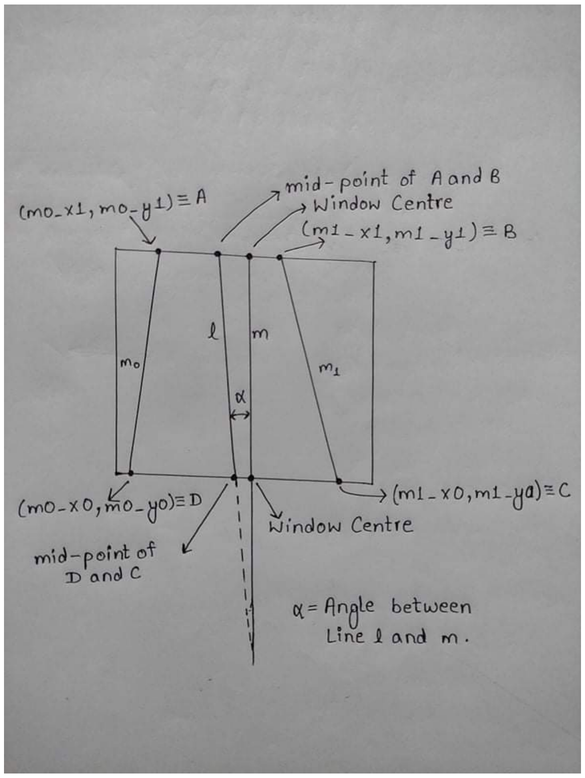

Figure 7: Angle Co-ordintes

Following methods can be adopted:

- Using the coordinates above, apply the midpoint according to the condition by calculating

$$\alpha = \arctan((52 - m)/(1 + 52 * m))$$

so that car moves along the main trajectory, where m is the slope of the line calculated by the midpoints and alpha is angle between m and the constant slope for trajectory.

- Instead of taking angle as an error one can also take the distance between window centre and the mid point of upper coordinates. This will also help the car to maintain itself on the trajectory by minimising the error by maintaining the distance between window centre and midpoint and changing its angular velocity accordingly.Such error can be resolved by using PID controller.

7 Pure pursuit algorithm

The pure pursuit algorithm was originally devised as a method for calculating necessary to get a robot back onto a path. The implementation of the pure pursuit algorithm itself is fairly straightforward. The pure pursuit algorithm can be outlined as follows:

- Determine the anent location of the vehicle.
- Find the path point closest to the vehicle.
- Find the goal point.
- Transform the goal points to vehicle coordinates.
- Calculate the curvature and request the vehicle to set the steering to that curvature.
- Update the vehicle's position.

8 Algorithm for PID controller

8.1 PID(Proportional, Integral, Derivative):

The goal of this project is to simulate self driving car for which a control scheme is used: PID controller algorithm for a car to drive around a track in the simulator while staying within the lane. This is accomplished by calculating the steering angle that is proportional to the Cross Track Error (CTE), which is the lateral distance between the car and the reference trajectory.

Proportional component: The proportional term, when used by itself to calculate the steering angle, sets a steering angle that is proportional to the CTE. However, the end result is a steering angle which oscillates around the reference trajectory. The proportional coefficient (Kp) determines how fast the car oscillates(or overshoots) around the reference trajectory.

Derivative component: The derivative component uses a rate of change of error to reduce the overshoot caused by the proportional component. by varying speed and kp, kd, ki in the algorithm, it is expected to move the car on desired path. hence the car now remains in middle of the path and moves in accelerated mannerThis derivative coefficient (Kd) term is used to optimize how far the car overshoots (also known as oscillation amplitude) from the reference trajectory.

Integral component: Over time, the steering angle accrues errors due to systematic bias which could drive the car out of the track eventually, but not immediately. The integral component fixes this problem. As this component impacts the error over a period of time, the integral coefficient (Ki) should be carefully optimized in small steps as it has a large impact on the overall performance.

$$PID = Kp*currenterror+Kd*(currenterror-previouserror)+Ki*s$$

Where:
s = summing last 10 errors obtained to multiply with Ki

$$currenterror = \frac{(a1 - windowcenter)}{50}$$

$$windowcenter = \frac{framewidth}{2}$$

$$framewidth = 100$$

$$a1 = \frac{(m0x1 + m1x1)}{2}$$

Therefore, by varying speed and kp, kd, ki in the algorithm, it is expected to move the car on desired path. hence the car now remains in middle of the path and moves in accelerated manner.

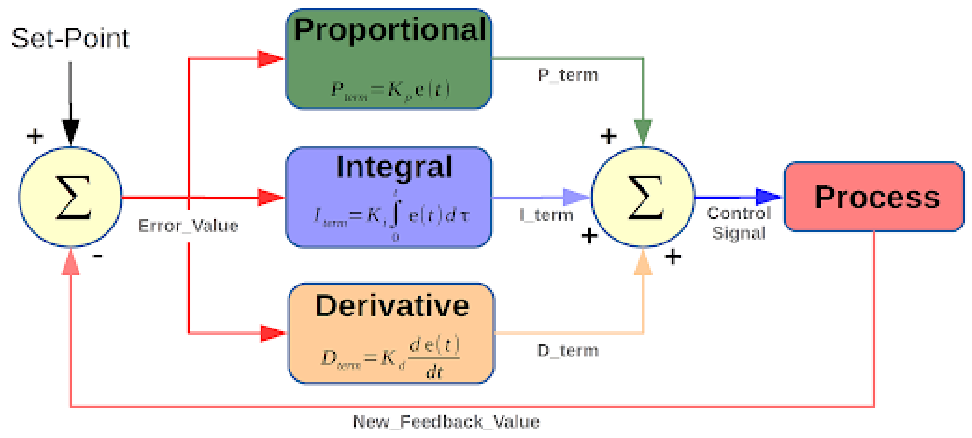

Figure 8: Roadmap of PID Controller

9 Future aspects

Above stated algorithm proves fairly sufficient to move the car on track in a balanced manner.

- It can also be modified to control car's motion on a path with obstacles.

- It can also be used to find shortest path by applying other algorithms.

- Stimulating algorithm from one track to another track and find out the best algorithm that can work on any track.

10 Github

https://github.com/Hardik12Jain/Autonomous-ROS.git