AUTONOMOUS VEHICLE

A PROJECT REPORT

Submitted by

R. MOHAMED ASHIK BADUSHA	**(412518106087)**
R. SIVAKUMAR	**(412518106159)**
K. BALAJI	**(412518106302)**

In partial fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

SRI SAIRAM ENGINEERING COLLEGE (AUTONOMOUS),
SAI LEO NAGAR, CHENNAI-44
ANNA UNIVERSITY: CHENNAI 600 025
MAY 2022

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“AUTONOMOUS VEHICLE”** is the bonafide work of **“R. MOHAMED ASHIK BADUSHA (412518106087), R. SIVAKUMAR (412518106159), K. BALAJI (412518106302),” who** carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. J. RAJA **PROFESSOR** **HEAD OF THE DEPARTMENT**	**Mr. K. DEVIBALAN** **ASSISTANT PROFESSOR** **SUPERVISOR**
Department of Electronics and Communication Engineering, **Sri Sai Ram Engineering College,** **(Autonomous) Chennai – 600044.**	Department of Electronics and Communication Engineering, **Sri Sai Ram Engineering College,** **(Autonomous) Chennai – 600044.**

Submitted for **VIVA-VOCE** Examination held on …………………...

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We thank our Founder Chairman **Thiru. MJF. Ln. LEO MUTHU** for his great endeavours in establishing this institution and standing as a figure of guidance.

We also thank our **CEO Mr. J. SAI PRAKASH LEO MUTHU** and Principal **Dr K PORKUMARAN** for their kind cooperation and inspiration.

We thank **Dr J. RAJA**, Professor, Head of the Department, Electronics and Communication Engineering for giving us the freedom to carry out the project work in the chosen domain.

We thank our Project Coordinators **Mr. U. JAYACHANDRAN**, Assistant Professor, **Dr DEEPA NIVEDITHA**, Assistant Professor, for their constant support right from the commencement of the project work and also for providing us necessary details with regard to presentation and documentation.

We are ever grateful to our Project Guide, **Mr. K. DEVIBALAN,** Assistant Professor, who was a buttress to carry out our project and for his valuable suggestions at every stage of the project. We sincerely thank him for the support rendered from the day we commenced our work.

Our gratitude extends to the **Teaching and Non-Teaching Staffs of the ECE Department** whose words of encouragement kept our spirits high throughout the course of the project.

We thank the entire people who contributed directly and indirectly for the completion of this project.

TABLE OF CONTENTS

ABSTRACT

An autonomous car, also referred to Self-driving cars 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 while you sit back, relax, and enjoying the smooth ride.

Objective of this project is to build Level 3 vehicles Which have “environmental detection” capabilities and can make informed decisions for themselves, such as accelerating past a slow-moving vehicle. But they still require human override. The driver must remain alert and ready to take control if the system is unable to execute the task. Autonomous driving is not one single technology, but rather a highly complex system that consists of many sub-systems. Let us break it into three major components namely algorithms including sensing, perception, and decision. Client systems, including the operating system and the hardware platform including high-definition (HD) Camera, deep learning trained model, simulation Code, and data storage. The algorithm subsystem extracts meaningful information from sensor raw data to understand its environment and to make decisions about its future actions. The client systems integrate these algorithms together to meet real-time and reliability requirements, in other words taking actions so as to reliably and safely reach target destinations.

Autonomous vehicle technology may be able to provide certain advantages compared to human-driven vehicles. One such potential advantage is that they could provide increased safety on the road. Vehicle crashes cause many deaths every year, and automated vehicles could potentially decrease the number of casualties as the software used in them is likely to make fewer errors in comparison to humans. A decrease in the number of accidents could also reduce traffic congestion, which is a further potential advantage posed by autonomous vehicles. Autonomous driving can also achieve this by the removal of human behaviours that cause blockages on the road, specifically stop-and-go traffic. Another possible advantage of automated driving is that people who are not able to drive due to factors like age and disabilities could be able to use automated cars as more convenient transport systems. Additionally, that come with an autonomous car are elimination of driving fatigue and being able to sleep during overnight journeys.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Autonomous cars are capable of driving with Minimal to no human input. A fully self-driving car will help you to drive from One place to another while you sit back, relax, and enjoying the smooth ride.

Self-driving cars have been receiving tremendous attention recently, due to the technological boom in Artificial Intelligence (AI). In just the past few years, AI has gone from being nearly forgotten to becoming the largest Research and Development (R&D) investment at many organizations across the world. Put simply, AI has given us the ability to automate a lot of manual work that would previously have taken some form of human knowledge or skill. In the case of self-driving cars, here, AI is used to help with being the brains of the cars doing things like automatically detecting people, cars around the vehicle, staying on the lane, switching lanes, and following the GPS and more on.

Automation of Vehicle can help reduce the number of crashes on our roads. Government data identifies driver behaviour or error as a factor in 94 percent of crashes, and self-driving vehicles can help reduce driver error. Higher levels of autonomy have the potential to reduce risky and dangerous driver behaviours. By use of Vision Based System which detects the speed limit signs or sign boards for schools/hospitals/accident prone areas etc. and may decide upon the safe speed value with which the vehicle need to operate. The Autonomous system will also provide a warning to the driver, in order to ensure the safety of the pedestrians. This project is mainly focused on Level 3 Automation. Level 3 Automation: Level 3 vehicles, also known as conditional driving automation vehicles, have “environmental detection” capabilities. As a result, drivers are able to engage in activities behind the wheel, such as using smartphones, under limited conditions while their car accelerates past a slow-moving vehicle or otherwise navigates expressway traffic.

2. LITERATURE SURVEY

2.1 AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles

Authors: Shital Shah, Debadeepta Dey, Chris Lovett, Ashish Kapoor

Purpose:

Developing and testing algorithms for autonomous vehicles in real world is an expensive and time-consuming process. Also, in order to utilize recent advances in machine intelligence and deep learning there is a need to collect a large amount of annotated training data in a variety of conditions and environments. Hence presenting a new simulator built on Unreal Engine that offers physically and visually realistic simulations for both of these goals. Our simulator includes a physics engine that can operate at a high frequency for real-time hardware in the loop (HITL) simulations with support for popular protocols (e.g., MavLink). The simulator is designed from the ground up to be extensible to accommodate new types of vehicles, hardware platforms and software protocols. In addition, the modular design enables various components to be easily usable independently in other projects. This also demonstrates the simulator by first implementing a quadrotor as an autonomous vehicle and then experimentally comparing the software components with real-world flights.

2.2 Autonomous vehicles: The future of automobiles

Authors: Rajasekhar MV, Anil Kumar Jaswal

Purpose:

Autonomous cars are the future smart cars anticipated to be driver less, efficient and crash avoiding ideal urban car of the future. To reach this goal automakers have started working in this area to realize the potential and solve the challenges currently in this area to reach the expected outcome. In this regard the first challenge would be to customize and imbibe existing technology in conventional vehicle to translate them to a near expected autonomous car. This transition of conventional vehicles into an autonomous vehicle by adopting and implementing different upcoming technologies is discussed in this paper. This includes the objectives of autonomous vehicles and their implementation difficulties. The paper also touches upon the existing standards for the same and compares the introduction of autonomous vehicles in Indian market in comparison to other markets. There after the acceptance approach in Indian market scenarios is discussed for autonomous vehicles.

2.3 Autonomous vehicles

Authors: Reza Langari, ASME Fellow

Purpose:

This summary presents the outline of a tutorial session on autonomous driving, which will include a detailed presentation by the author as well as a panel discussion by industry representatives from Mitsubishi and Emmeskay, Inc. The main presentation will address the fundamental issues in autonomous driving, namely i) localization, ii) perception, ii) decision logic, iv) control execution as well as v) validation/verification. Additional consideration will be given to ethical issues in autonomous driving. The state of the art in the field across these areas is presented and open research issues are highlighted. The industry panel (which may include those other than the aforementioned and are yet to be confirmed) will discuss their perspectives on these issues and present their case for further research and development in this emerging area.

2.4 Design & implementation of real time autonomous car by using image processing & IoT

Authors: Irfan Ahmad, Karunakar Pothuganti

Purpose:

Because of the inaccessibility of Vehicle-to Infrastructure correspondence in the present delivering frameworks, (TLD), Traffic Sign Detection and path identification are as yet thought to be a significant task in self-governing vehicles and Driver Assistance Systems (DAS) or Self Driving Car. For progressively exact outcome, businesses are moving to profound Neural Network Models Like Convolutional Neural Network (CNN) as opposed to Traditional models like HOG and so forth. Profound neural Network can remove and take in increasingly unadulterated highlights from the Raw RGB picture got from nature. In any case, profound neural systems like CNN have a highly complex calculation. This paper proposes an Autonomous vehicle or robot that can identify the diverse article in condition and group them utilizing CNN model and through this information can take some continuous choice which can be utilized in the Self Driving vehicle or Autonomous Car or Driving Assistant System (DAS).

3. IMPLEMENTATION AND DESIGN

3.1 Track Construction:

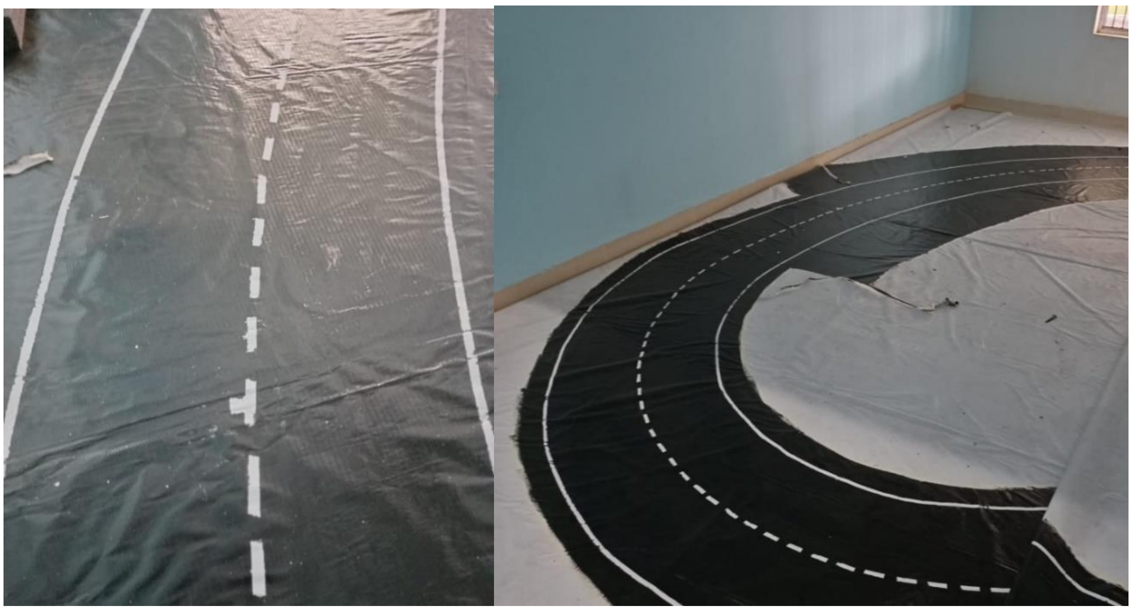


Fig 3.1 Lane construction

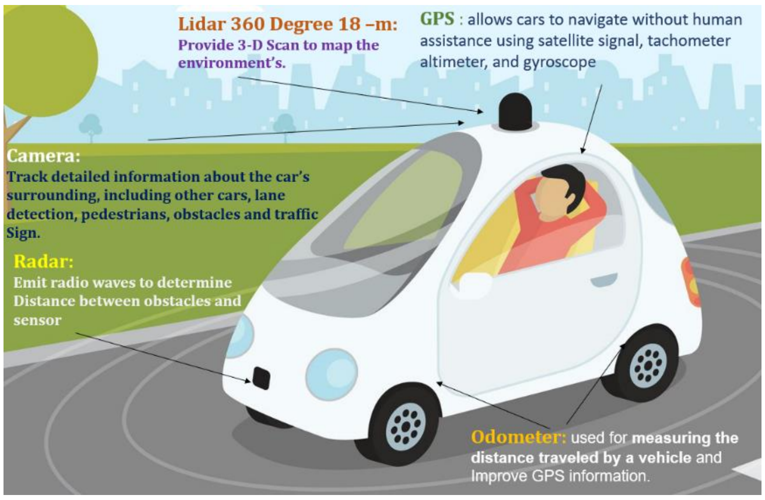
Based on the vehicle’s dimension, an artificial road has been constructed accordingly. The exact dimensions of the road are as follows:

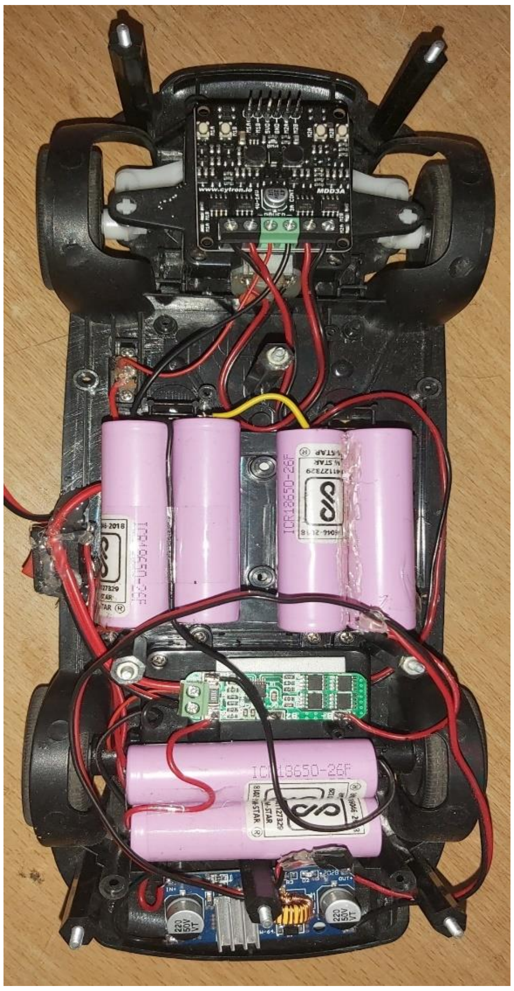
Measurements:

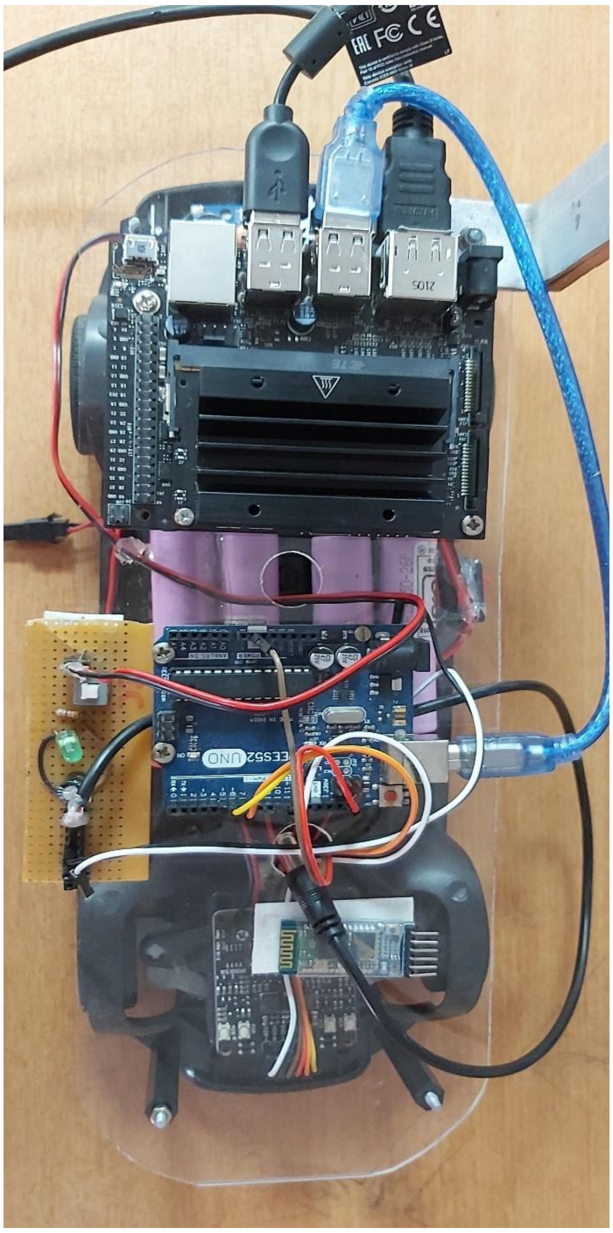
- Total road Width: 24 inches
- Right-side & Left-side Road width: 11 inches
- Lane Width: 0.5 inch
- Length of the road: 20 meters

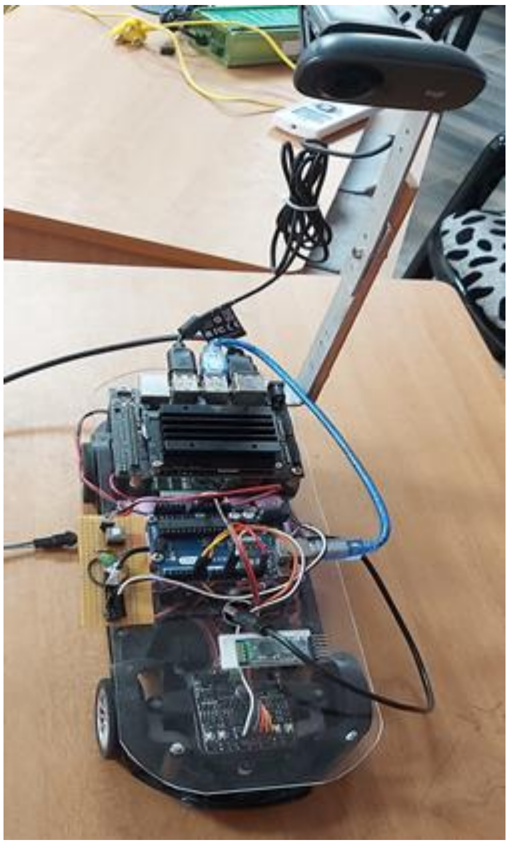
3.2 Car Development:

fig 3.2 Car Design


Lidar 360 Degree 18 –m:
Provide 3-D Scan to map the environment's.
GPS : allows cars to navigate without human assistance using satellite signal, tachometer altimeter, and gyroscope
Camera:
Track detailed information about the car's surrounding, including other cars, lane detection, pedestrians, obstacles and traffic Sign.
Radar:
Emit radio waves to determine Distance between obstacles and sensor
Odometer: used for measuring the distance traveled by a vehicle and Improve GPS information.







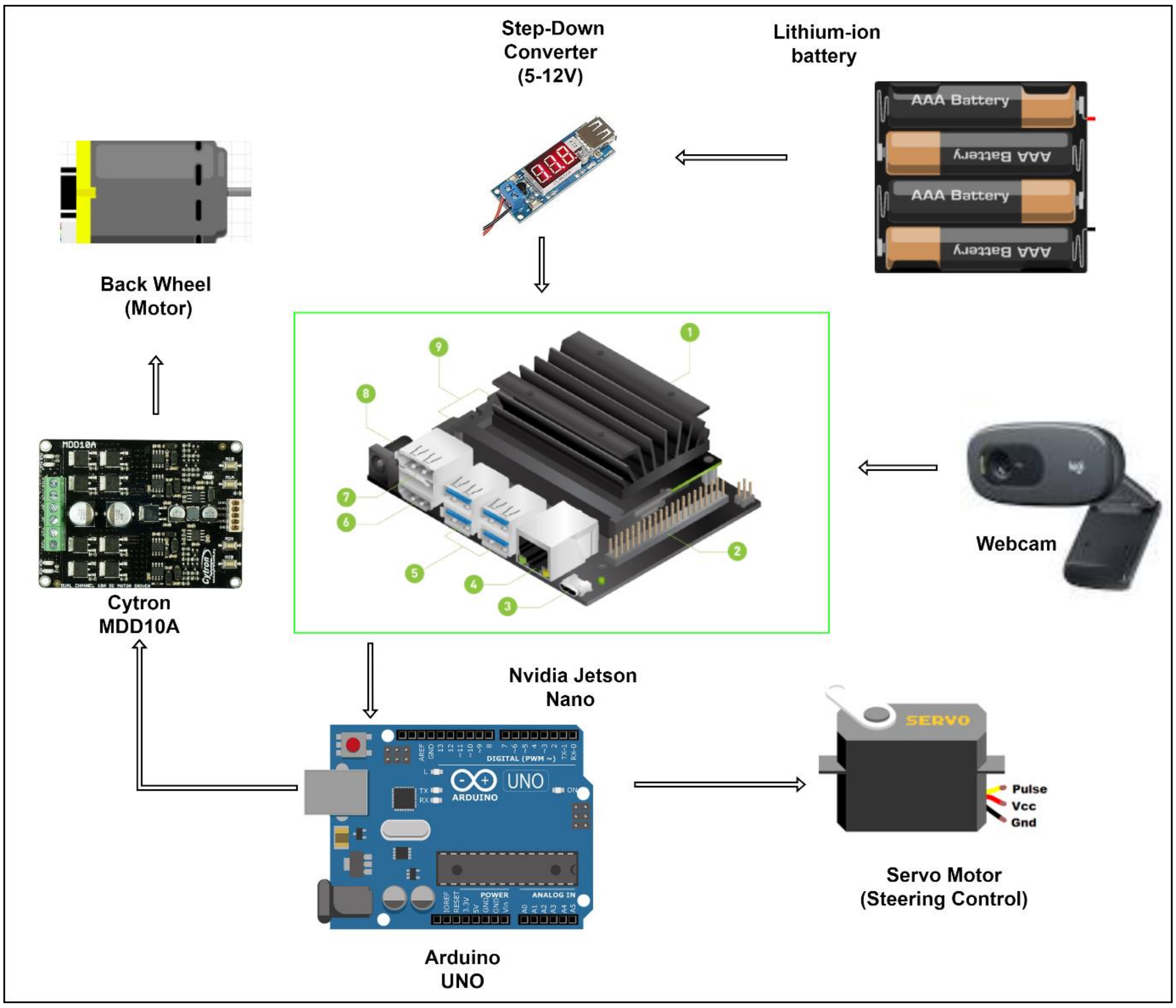


Fig 3.3 Block_diagram

A camera is placed in the bonnet of the vehicle, it will capture the real time image of the vehicle front view and process the image to detect the lanes and object in the image. Depending upon the output obtained, the main processor (Jetson) will make a decision by comparing the data obtained from the frame to the pre-processed data of the obstacle detected and intelligently plans what to do next and sends the result to the secondary controller (Arduino) which is responsible for controlling the servo motor which acts as a steering for front wheels, and to back wheel motors for front, reverse and stop movement.

3.3 Circuit Diagram:

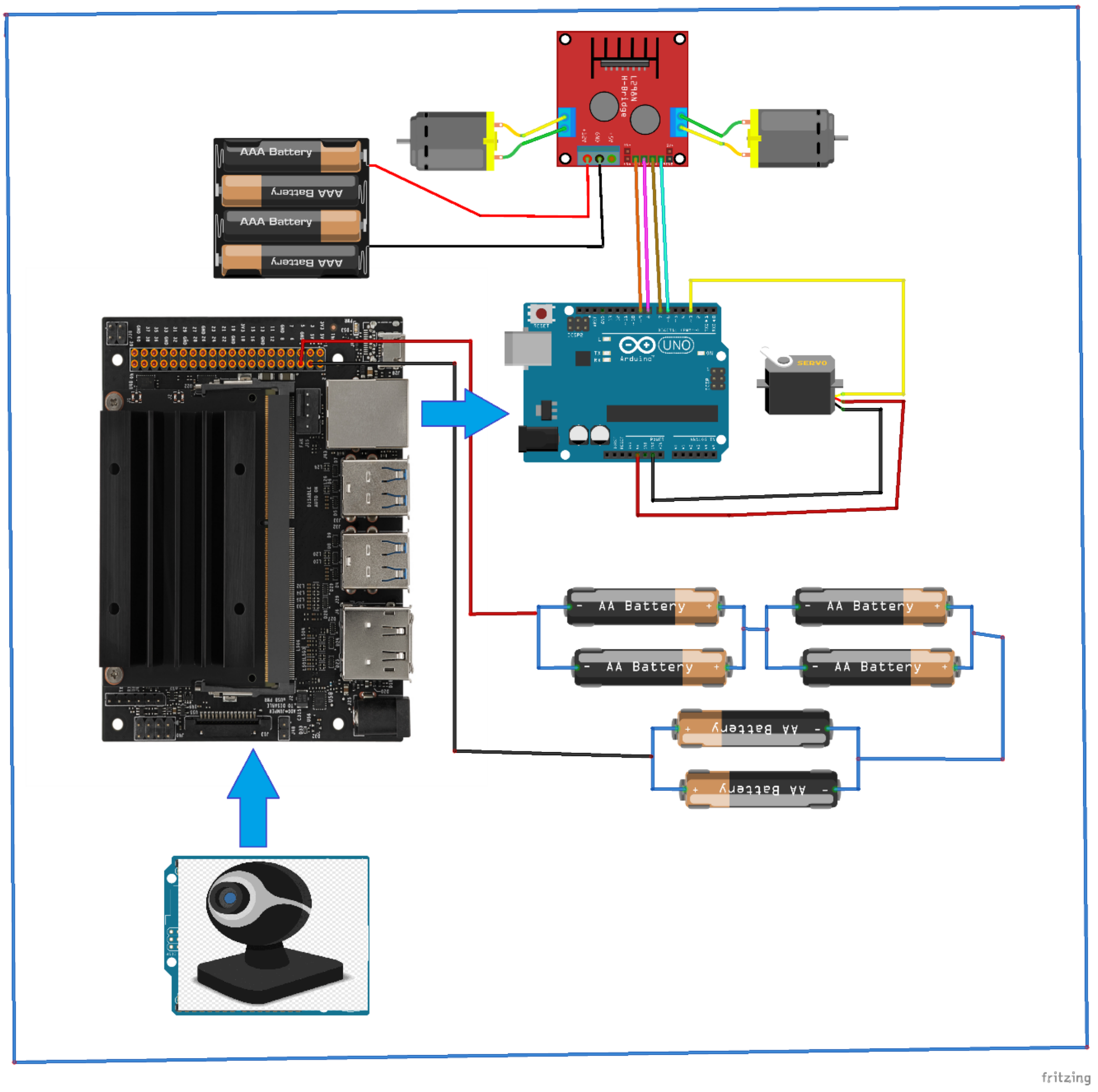


Fig 3.4 Circuit diagram

3.4 Lane Detection:

- Convert original image to grayscale.
- Darkened the grayscale image (this help in reducing contrast from discoloured regions of road)
- Convert original image to HLS colour space.
- Isolate white from HLS to get white mask. (for white lane markings)
- Bit-wise OR yellow and white masks to get common mask.
- Bit-wise AND mask with darkened image .

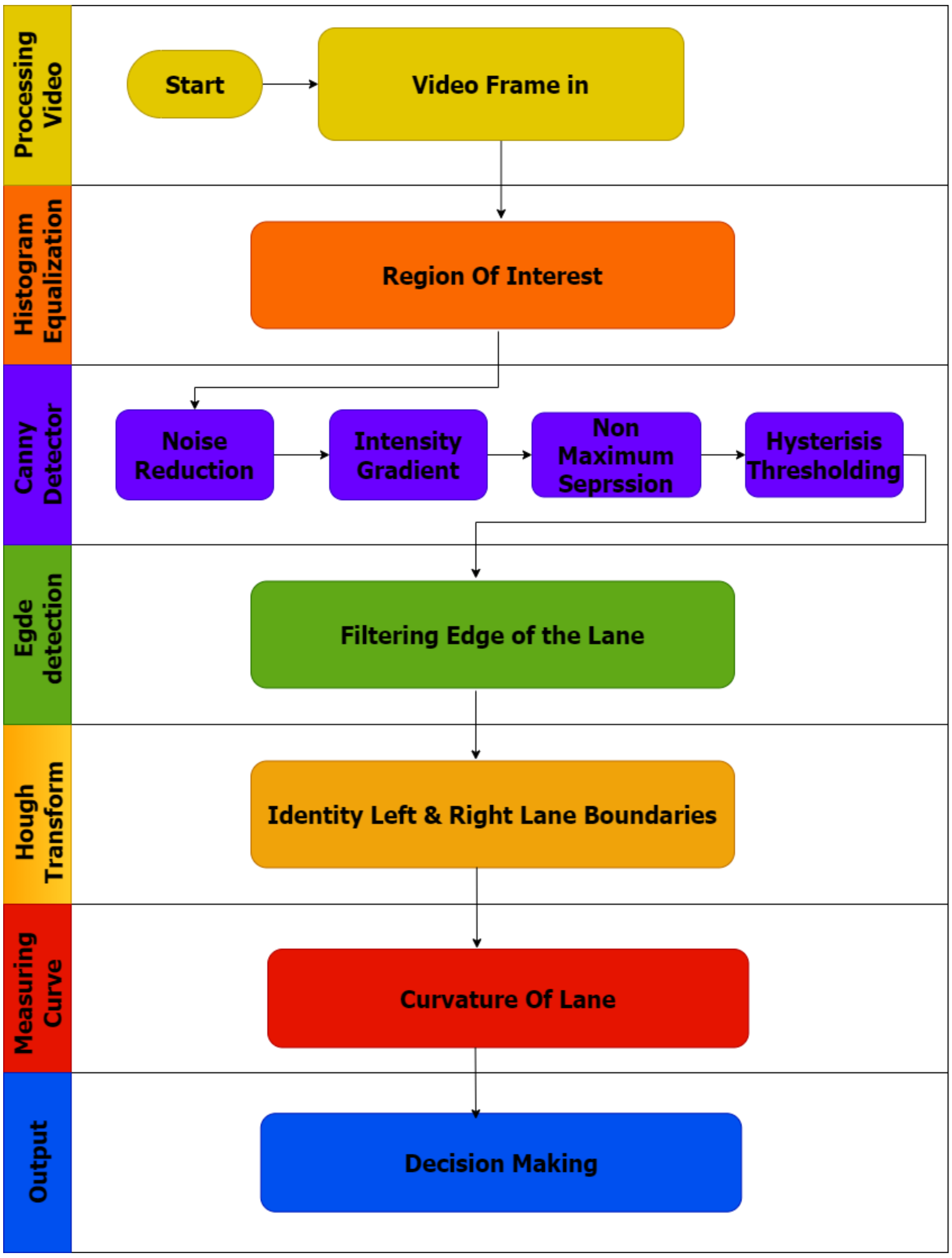


fig 3.5 Lane_detection_FlowChart

- Apply slight Gaussian Blur.
- Apply canny Edge Detector (adjust the thresholds — trial and error) to get edges.
- Define Region of Interest. This helps in weeding out unwanted edges detected by canny edge detector.
- Retrieve Hough lines.
- Consolidate and extrapolate the Hough lines and draw them on original image.

3.5 Jetson Nano Setup:

Initial setup:

1. Head over to Nvidia's official site to download Jetpack and follow the instructions to install it. You will need to download SD card formatter tool and jetpack image flashing tool. Everything will be mentioned in the steps.
2. After flashing jetpack to SD card, plug it in and also attach monitor, keyboard and mouse.
3. After booting it, increase the swap size to 4GB.Now connect your jetson to internet via ethernet cable or Intel's wifi/bluetooth module for jetson.
4. Now run Following Commad in terminal:
   ```
   sudo apt-get update
   sudo apt-get upgrade
   sudo apt install nano
   ```
5. Jetson Nano is now ready

Installing required libraries:

1. Remember to use pip3 while installing these libraries(you have python 2 and python 3 both installed). Anaconda is not supported by Jetson as per today. If you want to create virtual environments you can use python's venv or Anaconda.
2. Install numpy version 1.19.4 only. Any version above this can cause problems while using Pytorch or Tensorflow (Faced issues myself — tensorflow threw core dumped error). To install numpy 1.19.4. Also install opencv :-
   ```
   sudo apt-get install python3-pip
   pip3 install cython
   pip3 install numpy==1.19.4
   pip3 install opencv-python
   ```

3. Download few dependencies before installing Pytorch:-
```
sudo apt-get install python3-pip libjpeg-dev libopenblas-dev libopenmpi-dev libomp-dev
sudo apt install libfreetype6-dev python3-dev zlib1g-dev
```
4. Head over to https://forums.developer.nvidia.com/t/pytorch-for-jetson-version-1-9-0-now-available/72048 or search "download pytorch jetson" and open Nvidia's website. Now, click on arrow and download Pytorch 1.8. It will download a .whl file. Install it using:-
```
pip3 install path_to_pytorch18.whl
```
5. Pytorch also requires torchvision library. To install it, (in the below command, vision and torchvision are separated by space — copy the entire clone command till the word torchvision and paste it in the terminal)
```
git clone — branch v0.9.1 https://github.com/pytorch/vision torchvision
cd torchvision/
sudo python3 setup.py install
```
6. Now come out of the torchvision folder by using command cd .. and verify installation of Pytorch and torchvision by importing them. If you see this error "Cannot find Python.h" while installing torchvision, kindly reinstall python3-dev package.
7. Now install Tensorflow 2.4.1 by following this article — https://qengineering.eu/install-tensorflow-2.4.0-on-jetson-nano.html
8. If gdown command(used to download tf2.4.1.whl file) doesn't work, then simply copy the drive link from the steps and download the file yourself.
9. Verify the installation of tensorflow by importing it and printing the version.

Running Yolov5 on your Jetson:

1. Clone the repo and move inside the cloned folder
```
git clone https://github.com/ultralytics/yolov5.git
cd yolov5
export OPENBLAS_CORETYPE=ARMV8
```
2. Now type nano requirements.txt in terminal and comment out these packages as these are already installed -> numpy, tensorflow, torch, torchvision and opencv
3. Now install the remaining libraries using the following command. This process may take time as many of these libraries are downloaded and

built locally on the jetson.

```
pip3 install -r requirements.txt
```

4. After installing, download Yolov5s model from the repository, place it in the yolov5 directory and run the following. If the following commands throws error, kindly look for missing dependencies.

   ```
   python3 detect.py --weights yolov5s.pt --source 0 --device 0
   ```

5. Source 0 means you are using Webcam. (change the source to path of image or video if you don’t want to use live feed). Device 0 means that the model should be loaded in GPU.
6. Do checkout detect.py for more options while running inference on YoloV5s model (Yolov5s is a light weight model)
7. Jetson Nano is ready for Execution.

4. HARDWARE AND SOFTWARE

4.1 COMPONENTS USED

4.1.1 HARDWARE:

- ❑ Nvidia Jetson Nano
- ❑ Arduino UNO
- ❑ Step-Down Buck Converter (Module KG235)
- ❑ Motor Driver - Cytron MDD10A
- ❑ Speed Measuring Sensor HC-020K
- ❑ TF02-Pro LIDAR (40M)
- ❑ Logitech C270 HD Webcam
- ❑ Lithium-ion battery
- ❑ Servo Motor - MG995

4.1.2 SOFTWARE

- ❑ Python (version=3.7.10)
- ❑ OpenCV (version=4.1.1)
- ❑ ROS 2
- ❑ TensorFlow (version=2.4.1)
- ❑ Pytorch (version=1.8.1)
- ❑ YOLO v5s or SSD MobileNet-v2
- ❑ Jupyter notebook
- ❑ Anaconda Distribution
- ❑ NVIDIA Cuda and Cudnn

4.2 COMPONENTS DISCRIPTION

4.2.1 Jetson

NVIDIA Jetson is a platform for embedded and edge devices that combines high-performance, low-power compute modules with the NVIDIA AI software stack. It's designed to provide end-to-end acceleration for AI applications with the same NVIDIA technologies that power data center and cloud deployments.

The Jetson family of production modules and developer kits include

- Jetson Orin
- Jetson Xavier
- Jetson Nano
- Jetson TX1
- Jetson TX2
- Jetson TK1

Jetson Modules

Each Jetson module is a complete System-on-Module (SOM) with NVIDIA GPU, CPU, memory, PMIC, etc. The modules also include AI, Computer Vision, and other hardware accelerators.Jetson modules are designed for deployment in a production environment throughout their operating lifetime. Each Jetson module ships with no software pre-installed; you attach it to a carrier board designed or procured for your end product, and flash it with the software image you've developed.

Jetson Software

NVIDIA Jetson production modules and developer kits are all supported by the same **NVIDIA software stack**, enabling you to develop once and deploy everywhere. **JetPack SDK** includes the latest Jetson Linux Driver Package (L4T) with Linux operating system and CUDA-X accelerated libraries and APIs for AI Edge application development. It also includes samples, documentation, and developer tools for both host computer and developer kit, and supports higher level SDKs such as Deep Stream for streaming video analytics and Isaac for robotics.

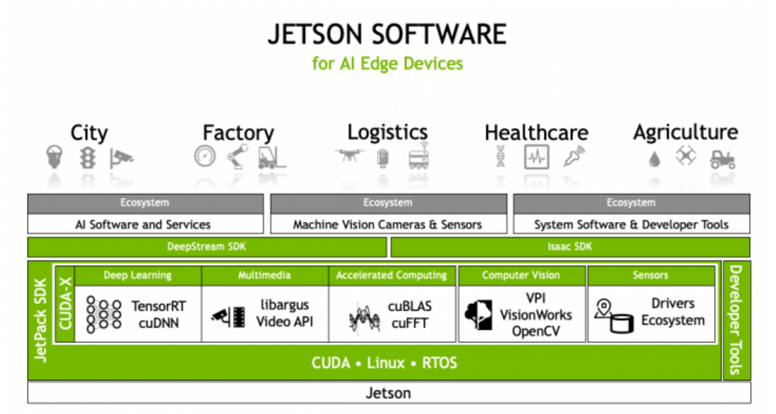


Fig 4.1 Jetson_Software

Jetson Nano

NVIDIA ***Jetson Nano*** is an embedded system-on-module (SoM) and developer kit from the *NVIDIA Jetson* family, including an integrated 128-core Maxwell GPU, quad-core ARM A57 64-bit CPU, 4GB LPDDR4 memory, along with support for MIPI CSI-2 and PCIe Gen2 high-speed I/O. There is also the ***Jetson Nano 2GB Developer Kit*** with 2GB memory and the same processing specs.Jetson Nano runs Linux and provides 472 GFLOPS of FP16 compute performance with 5-10W of power consumption.

Jetson Nano Developer Kit

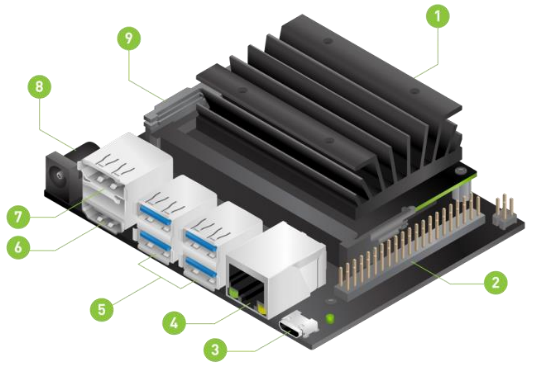
What's Included

- 80x100mm Reference Carrier Board
- Jetson Nano Module with passive heatsink
- Pop-Up Stand

What You Will Need

- Power Supply
  - Jetson Nano Developer Kit *(Micro-USB or DC barrel jack)*
    - 5V⎓2.5A Micro-USB adapter (see *Adafruit GEO151UB*)
    - 5V⎓4A DC barrel jack adapter, 5.5mm OD x 2.1mm ID x 9.5mm length, center-positive (see *Adafruit 1446*)
    - See the *Jetson Nano Supported Component List* and *Power Supplies* section below for more information.
  - Jetson Nano 2GB Developer Kit *(USB-C power supply)*
    - 5V⎓3A USB-C adapter
    - See the *Jetson Nano 2GB Supported Component List* and *Power Supplies* section below for more information.
- MicroSD card (32GB UHS-1 recommended minimum)

Fig 4.2 Jetson_Config



1. microSD card slot for main storage
2. 40-pin expansion header
3. Micro-USB port for 5V power input or for data
4. Gigabit Ethernet port
5. USB 3.0 ports (x4)
6. HDMI output port
7. DisplayPort connector
8. DC Barrel jack for 5V power input
9. MIPI CSI camera connector

Ports & Interfaces

	Jetson Nano Developer Kit	*Jetson Nano 2GB Developer Kit*
USB	(4x) USB 3.0 Type-A, USB 2.0 Micro-B	(1x) USB 3.0 Type-A, (2x) USB 2.0 Type-A, USB 2.0 Micro-B
Camera	(2x) MIPI CSI-2 x2 (15-position Camera Flex Connector)	(1x) MIPI CSI-2 x2 (15-position Camera Flex Connector)
Display	HDMI 2.0, DisplayPort	HDMI 2.0
Wireless	M.2 Key-E (PCIe x1)	802.11ac USB dongle*
Ethernet	Gigabit Ethernet (RJ45)	
Storage	MicroSD card slot	
Other	40-pin Header - (3x) I2C, (2x) SPI, UART, I2S, GPIOs	
Power	Micro-USB (5V⎓2.5A) or DC barrel jack (5V⎓4A)	USB-C (5V⎓3A)

Table 4.1 jetson Ports & Interfaces

Software Support

- JetPack
- Linux4Tegra (L4T)
- Linux kernel 4.9
- Ubuntu 18.04 aarch64
- CUDA Toolkit
- cuDNN
- TensorRT
- TensorFlow
- VisionWorks
- NVIDIA Nsight Graphics
- NVIDIA Nsight Compute
- NVIDIA Nsight Systems
- Ubuntu 18.04 aarch64
- CUDA Toolkit
- cuDNN
- TensorRT
- TensorFlow
- VisionWorks
- OpenCV 4.1
- OpenGL 4.6 / OpenGL ES 3.2.5
- Vulkan 1.1.1
- L4T Multimedia API (Argus 0.97)
- GStreamer 1.14.1
- V4L2 media controller support

4.2.2 Arduino Uno

The **Arduino Uno** is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is similar to the Arduino Nano and Leonardo. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

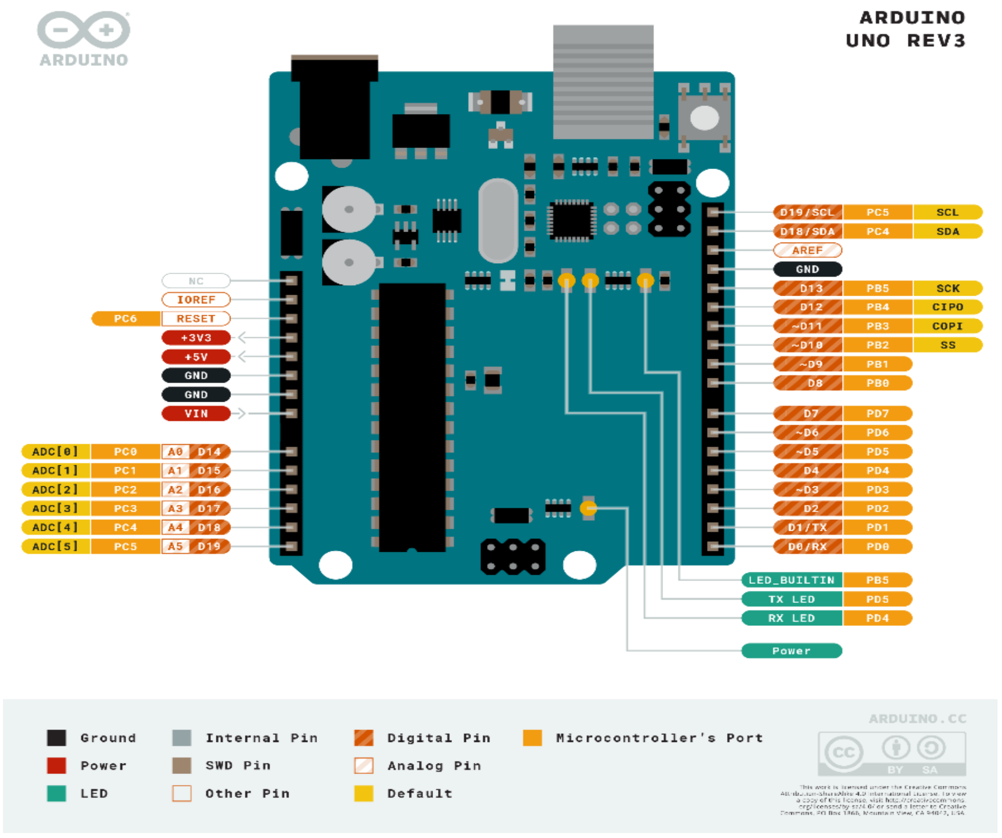


fig 4.3 Arduino_pin_diagram

Technical specs

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Table 4.2 Arduino_pins

Programming

The Arduino Uno can be programmed with the (Arduino Software (IDE)). The ATmega328 on the Arduino Uno comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then reseing the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

Input and Output

See the mapping between Arduino pins and ATmega328P ports. The mapping for the Atmega8, 168, and 328 is identical.

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e., 1024 different values). By default, they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function.
There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

4.2.3 Motor Driver - Cytron MDD10A

This Cytron Dual Channel Enhanced 10Amp DC Motor Driver 30A Peak MDD10A is the dual-channel version which is designed to drive 2 brushed DC motor with high current up to 13A continuously. Like MD10C, MDD10A also supports locked-antiphase and sign-magnitude PWM signal.

MDD10A is the dual channel version of MD10C which is designed to drive 2 brushed DC motors with high current up to 10A continuously. Just like MD10C, the MDD10A also supports locked-antiphase and sign-magnitude PWM signal. It is also using full solid-state components which result in faster response time and eliminate the wear and tear of the mechanical relay.

MDD10A has been designed with the capabilities and features of:

- Bi-directional control for 2 brushed DC motors.
- Support motor voltage ranges from 5V to 25V30V (Rev2.0).
- Maximum current up to 10A continuous and 30A peak (10 second) for each channel.
- Solid state components provide faster response time and eliminate the wear and tear of mechanical relay.
- Fully NMOS H-Bridge for better efficiency and no heat sink is required.
- Speed control PWM frequency up to 20KHz (Actual output frequency is same as input frequency).
- Support both locked-antiphase and sign-magnitude PWM operation.
- Support TTL PWM from microcontroller, not PWM from RC receiver .
- Onboard push button to control the motor manually.
- Dimension: 84.5mm x 62mm

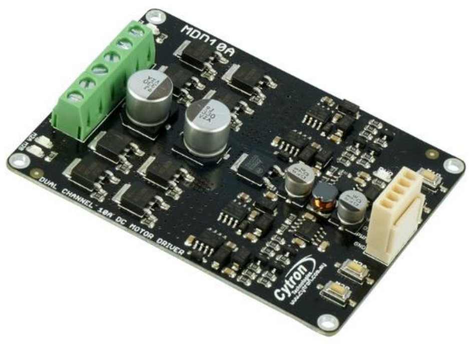


fig 4.4 Cytron MDD10A

Features:

1. Bi-directional control of 2 brushed DC motor.
2. Support motor voltage ranges from 5V to 25V.
3. Maximum current up to 10A continuous and 30A peak (10 seconds) for each channel.
4. Solid-state components provide faster response time and eliminate the wear and tear of the mechanical relay.
5. Fully NMOS H-Bridge for better efficiency and no heat sink is required.
6. Speed control PWM frequency up to 20KHz. Support both locked-antiphase and sign-magnitude PWM operation.
7. 2 activation buttons for fast test on each channel.

Specifications:

Arduino Sheild	No- can be used with Wire connection
Operating Modes	PWM (Lock-Antiphase and Sign-Magnitude): Yes, UART: No, Analog:No, RC Servo Signal: No
Continuous Current (A)	10
Cooling Fan	No
Driver Model	MDD10A
LED Indicator	Yes
No. of Channels	2
Operating Voltage (VDC)	5 ~ 25
Peak Current (A)	30
Polarity Protection	NO
Dimensions in mm (LxWxH)	85 x 62 x 25
Weight (gm)	40
Others	No heat sink is required
Shipment Weight	0.043 kg
Shipment Dimensions	7 × 5 × 3 cm

Table 4.3 Motor Driver Spec

4.2.4 Servo Motor – MG995

The TowerPro MG995 High-Speed Digital Servo Motor rotates 90° in each direction making it 180° servo motor. It is a Digital Servo Motor that receives and processes PWM signal faster and better. It equips sophisticated internal circuitry that provides good torque, holding power, and faster updates in response to external forces. These TowerPro MG995 Sem-Metal Gear Servo Motors are high-speed servo motors with a mighty torque of 10 kg/cm.

Wire Description

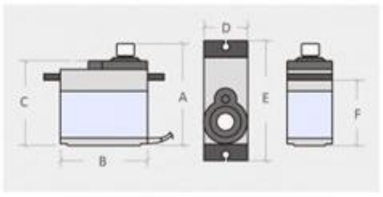
RED – Positive

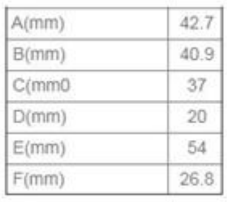
Brown – Negative

Orange – Signal

Features:

1. The connection cable is thicker.
2. Equips high-quality motor.
3. High resolution.
4. Accurate positioning.
5. Fast control response.
6. Constant torque throughout the servo travel range.





A(mm)	42.7
B(mm)	40.9
C(mm0	37
D(mm)	20
E(mm)	54
F(mm)	26.8

fig 4.5 Servo_Config

Specifications:

Model	MG995
Weight(gm)	55
Operating Voltage (VDC)	4.8 ~ 7.2
Operating Speed @4.8V	0.20sec/60°
Operating Speed @6.6V	0.16sec/60°
Stall Torque @ 4.8V (Kg-Cm)	10
Stall Torque @6.6V (Kg-Cm)	12
Operating Temperature (°C)	-30 to 60
Dead Band Width (µs)	1
Gear Type	Semi-Metal
Rotational Degree	180º
Servo Plug	JR
Cable Length (cm)	30
Length (mm)	40.5
Width (mm)	20
Height (mm)	44
Shipment Weight	0.059 kg
Shipment Dimensions	9 × 8 × 6 cm

Table 4.4 Servo Spec

4.2.5 Logitech C270 HD Webcam:

The Logitech C270 HD streams and records at 720p/30fps. The lens is plastic instead of glass, and it still produces a *relatively* clear picture. However, it doesn't compare to the quality of a 1080p webcam. You need to have realistic expectations regarding the type of image that a 720p webcam can produce.

The webcam has a fixed focus as opposed to autofocus. In addition, it doesn't include many of the features found in other Logitech webcams, such as pan, tilt, and zoom. It should also be noted that you can't rotate it from landscape to portrait mode, and the field of view is fixed at 60 degrees. But for many people, these features aren't necessary.

DIMENSIONS

Dimensions including fixed mounting clip

- Height: 72.91 mm
- Width: 31.91 mm
- Depth: 66.64 mm
- Cable length: 1.5 m
- Weight: 75 g

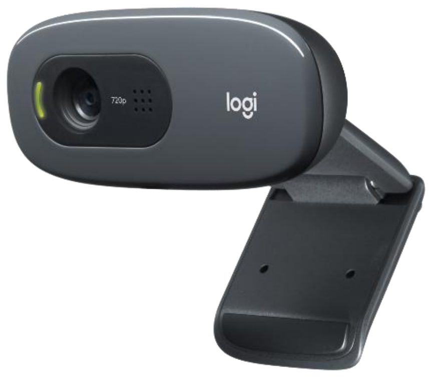


fig 4.6 Webcam

SYSTEM REQUIREMENTS

Compatible with

- Windows® 8 or later
- macOS 10.10 or later
- Chrome OS™
- USB - A port

- Works with popular calling platforms.

TECHNICAL SPECIFICATIONS

Max Resolution: 720p/30fps

Camera mega pixel: 0.9

Focus type: fixed focus

Lens type: plastic

Built-in mic: Mono

Mic range: Up to 1 m

Diagonal field of view (dFoV): 55°

Universal mounting clip fits laptops, LCD or monitors.

4.2.6 Speed Measuring Sensor HC-020K:

HC-020K speed measuring sensor is a wide voltage, high resolution, short response time, and the switch output speed measurement module. It can test motor's rotational speed with black encoder (measured spec. is related to the encoder, the inner diameter of D type encoder that provided is 4mm, can be used for motor output shaft w/ 4mm diameter, which is TT motor that is matched, yellow shell and white axis).

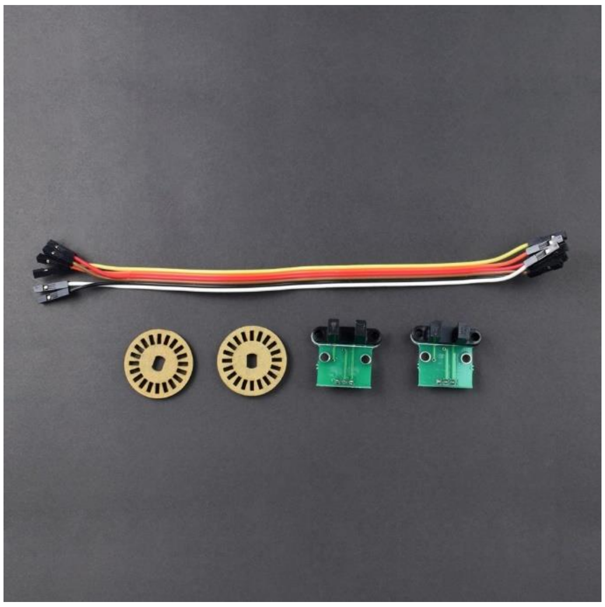


fig 4.7 Encoder

Specification:

- Module Working Voltage: 4.5-5.5V
- Launch Tube Pressure Drop: Vf=1.6V
- Launch Tube Current: If<20mA
- Signal output: A, B two lines; TT power level
- Resolution: 0.01mm
- Measurement frequency: 100KHz
- Disc diameter: 24mm
- Inner Disc Diameter: 4mm
- Encoder resolution: 20 lines
- Speed measuring sensor configuration: measure line 1 motor speed

Features:

- Wide voltage, high resolution, short response time, switch output.
- Can test motor's rotational speed w/ black encoder.

4.2.7 Lithium-ion battery:

A lithium-ion battery or Li-ion battery is a type of rechargeable battery composed of cells in which lithium ions move from the negative electrode through an electrolyte to the positive electrode during discharge and back when charging. Li-ion cells use an intercalated lithium compound as the material at the positive electrode and typically graphite at the negative electrode. Li-ion batteries have a high energy density, no memory effect (Other than LFP cells) and low self-discharge. Cells can be manufactured to either prioritize energy or power density.

18650 battery is a Li-ion rechargeable battery with 2000 mAh Battery Capacity. This is not a standard AA or AAA battery but is very useful for applications which require continuous high current or high current in short bursts like in cameras, DVD players, iPod, etc. 18650 cells can be charged and discharged up to 1000 cycles without much loss in battery capacity. They are safe to use, environment friendly and has a long battery life. It comes with high energy density and provides excellent continuous power sources to your device. It should be used with a protection circuit board which guards the battery against over-charge, over-discharge of the pack, and avoid over-current drawn.

SPECIFICATIONS

- Voltage: 3.7 Volts
- Capacity: 2000 mAh
- Rechargeable: Yes
- Battery Size: Diameter- 18mm x Length- 67mm
- Charging Method CC-CV

fig 4.8 Lithium-ion battery

APPLICATIONS

- Cordless Phones
- Small DRONES
- Tablet PC GPS
- iPod, DVD, MP4 Player, etc.
- Mobiles backup power supply/Power bank

4.2.8 Step-Down Buck Converter (Module KG235):

The DC-DC 4.5-40V To 5V 2A USB Charger Step down Converter Voltmeter Module is made of electronic accessories, surface nickel plating (anti-rust), inner conductor: pure copper. It is long-lasting and durable. The module converts an input of 4.5-40V into a very stable 5V 2A output. The device features a USB port through which you can charge your phone. While charging your device please consider that the output of this device is regulated at 5V 2A.

Integrated voltmeter, buck voltage converter and 5V USB charger port, multifunctional and practical. Wide voltage measuring /input range: DC 4.5-40V; with self-calibration, high accuracy. Output Enable" button for USB output and "low power" button for low power-consumption mode. With reverse connection, over-current and over-temperature protection, hence safe for use. With LED indicator for USB power supply and wiring terminals, easy and convenient operation.

Simply connect this USB Charger Step down Converter Voltmeter Module device to the power supply, the voltage can be detected in real-time. When you need the phone, IPAD, MP3, and other digital products to charge, click the “Output Enable” button, then the USB power indicator lights up and it will start supplying 5V 2A at its output.

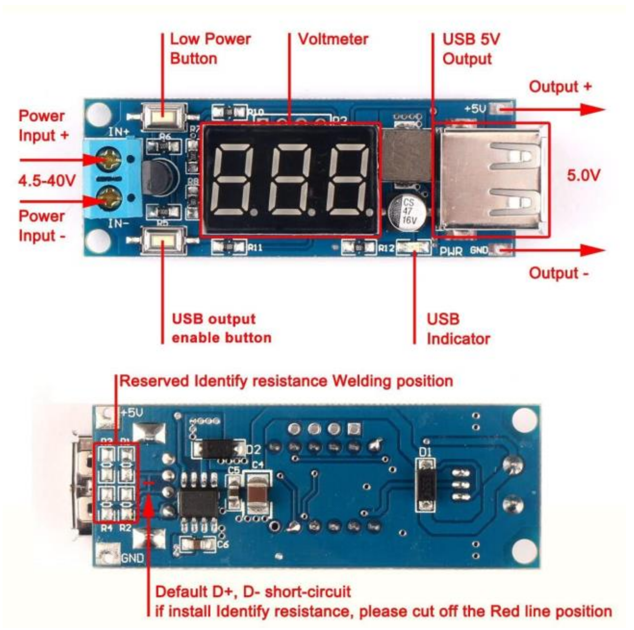

Low Power Button
Voltmeter
USB 5V Output
Output +
Power Input +
4.5-40V
Power Input -
5.0V
Output -
USB output enable button
USB Indicator
Reserved Identify resistance Welding position
Default D+, D- short-circuit
if install Identify resistance, please cut off the Red line position

fig 4.9 step_down_module

When you need to turn off the voltmeter, click the “Low Power” button, then enter the ultra-low-power standby mode, USB output corresponding shutdown. This DC-DC Buck Voltage Converter accepts wide input voltage as from car/battery car/motorcycle used for monitoring battery voltage; measure the voltage needed for a variety of occasions. The output current capability is 5.0V/2A, enough to meet a variety of USB devices.

Technical Details

- Input Voltage (V): 4.5 to 40
- Output Voltage(V): 5
- Max. Output Current (A): 2

Physical Attributes

- Length × Width × Height(mm): 61 × 11 × 13
- Weight (gm): 13

Application

- Cars and Electric Bikes.
- Motorcycle for voltage measurement and to charge the phone.
- iPads, Mobile Phones.
- Mp3 Devices.

Additional Details

- The voltmeter has an automatic calibration function.
- With reverse connection protection.
- With overheat protection and short circuit protection.
- High efficiency and low ripple.
- With USB power indicator light.

4.2.9 TensorFlow (version==2.4.1):

fig 4.10 TensorFLow

TensorFlow is an open-source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, TensorFlow was developed by the Google Brain team for internal Google use in research and production.] The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

TensorFlow can be used in a wide variety of programming languages, most notably Python, as well as JavaScript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors.

TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as *tensors*.

Tensor processing unit (TPU)

A TPU is a programmable AI accelerator designed to provide high throughput of low-precision arithmetic (e.g., 8-bit), and oriented toward using or running models rather than training them. Google announced they had been running TPUs inside their data centers for more than a year, and had found them to deliver an order of magnitude better-optimized performance per watt for machine learning.

4.2.10 Python (version==3.7.10)

fig 4.11 Python

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small- and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

Notable changes in Python 3.7.10

Earlier Python versions allowed using both; and & as query parameter separators in urllib.parse.parse_qs() and urllib.parse.parse_qsl(). Due to security concerns, and to conform with newer W3C recommendations, this has been changed to allow only a single separator key, with & as the default. This change also affects cgi.parse() and cgi.parse_multipart() as they use the affected functions internally. For more details, please see their respective documentation. (Contributed by Adam Goldschmidt, Senthil Kumaran and Ken Jin in bpo-42967.)

4.2.11 OpenCV (version==4.1.1):

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

Applications of OpenCV: There are lots of applications which are solved using OpenCV, some of them are listed below

- face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anamoly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

OpenCV Functionality

- Image/video I/O, processing, display (core, imgproc, highgui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)

- CUDA acceleration (gpu)

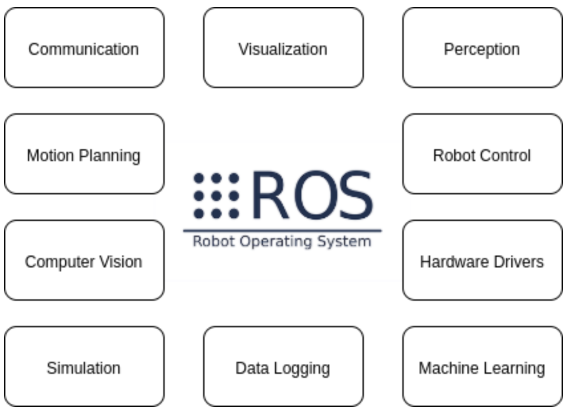
Image-Processing
Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it. Talking about the basic definition of image processing then **"Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality".**

Digital-Image:
An image may be defined as a two-dimensional function f(x, y), where x and y are spatial(plane) coordinates, and the amplitude of fat any pair of coordinates (x, y) is called the intensity or grey level of the image at that point. In another word an image is nothing more than a two-dimensional matrix (3-D in case of colored images) which is defined by the mathematical function f(x, y) at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what color it should be. Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirement associated with that image.
Image processing basically includes the following three steps:

1. Importing the image
2. Analyzing and manipulating the image
3. Output in which result can be altered image or report that is based on image analysis

4.2.12 ROS 2

fig 4.12 Ros_Config



Robot Operating System (ROS) is a set of open-source algorithms, hardware driver software and tools developed to develop robot control software. Even though it has operating system in its name it is not an operating system. It is

- Communication System (Publish Subscribe and Remote Method Invocation),

- Framework & Tools (Build system & dependency management, Visualization, Record and Replay)
- Ecosystem (Language bindings, Drivers, libraries and simulation (Gazebo)).

The first version of ROS was mostly used in academic projects. ROS 2 was developed so that it can be used in commercial projects. The biggest change that came with ROS2 was the selection of the DDS middleware for the communication layer. The DDS middleware, which has proven its worth in defense industry projects, has strengthened the communication between robot components with its decentralized publish subscribe architecture. Currently, ROS2 has been adapted 3–4 different DDS products. Since ROS2 is open source, related companies have also offered DDS libraries as open source to the sector. At the same time, ROS 2 has found a wider area of use with multi-robot communication, real-time communication and different platform support. Currently, ROS 2 is used in different areas from humanoid robots to industrial robots and autonomous vehicles.

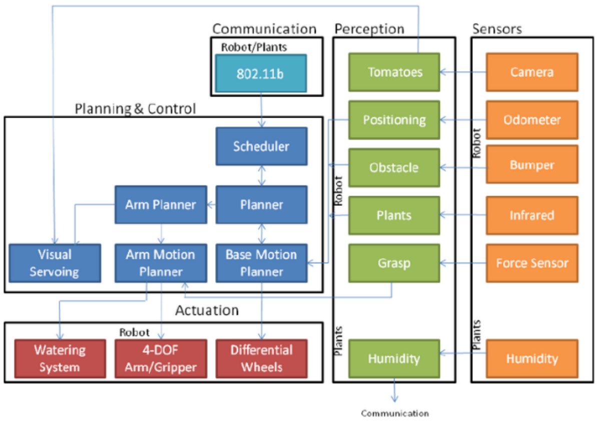
ROS includes mature open-source libraries to be used for navigation, control, motion planning, vision and simulation purposes. The 3D visualization tool called RVIS is an important tool used with ROS. Similarly, the simulation tool called Gazebo is seen as a useful tool for robot developers. Apart from this, Open CV library is a library used for detection purposes in ROS 2. In addition, see that the QT graphic library is also used for the user interface in ROS 2 projects and is an add-on available for this purpose.

ROS 2 Architecture

The ROS 2 has distributed real-time system architecture. Sensors in robots, motion controllers, detection algorithms, artificial intelligence algorithms, navigation algorithms, etc are all components (called as node) of this distributed architecture. The DDS middleware selected with ROS 2 for data exchange enables these components to communicate with each other in a distributed environment.

ROS 2 has provided its own abstraction layer (rmw) on top of DDS instead of directly using the DDS middleware. Thus, the details of the DDS middleware interface are abstracted from the user. In the current ROS 2 versions, Fast-DDS comes as the standard DDS version. Apart from that, rmw support is available in different DDS products. The user can select and use the DDS library they want, and even thanks to the network level interoperability, it is possible to use more than one DDS library in the same project.

ROS 2 Client Library



ROS 2 applications access ROS 2 features through the ROS Client Library (RCL) client library. The Rcl library is written in C language and on it there are Rclcpp client libraries for C ++ language and Rclpy for Python language. There are independently written ROS 2 client libraries in other languages such as Java, Go. The client library is primarily provided with the standard interface required to exchange data with Topic and Service approaches over ROS 2. In addition, the ROS2 library has capabilities to provide operating system abstraction and ready-made micro-architectural structures.

4.2.13 Pytorch (version==1.8.1):

fig 4.13 pytorch

PyTorch is an open-source machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab (FAIR). It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface.

A number of pieces of deep learning software are built on top of PyTorch, including Tesla Autopilot, Uber's Pyro, Hugging Face's Transformers, PyTorch Lightning, and Catalyst.

PyTorch provides two high-level features:

Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU).Deep neural networks built on a tape-based automatic differentiation system

PyTorch tensors

PyTorch defines a class called Tensor (torch.Tensor) to store and operate on homogeneous multidimensional rectangular arrays of numbers. PyTorch Tensors are similar to NumPy Arrays, but can also be operated on a CUDA-capable Nvidia GPU. PyTorch supports various sub-types of Tensors.

Note that the term "tensor" here does not carry the same meaning as in mathematics or physics. The meaning of the word in those areas is only tangentially related to the one in Machine Learning. In mathematics, a tensor is a certain kind of object in linear algebra, while in physics the term "tensor" usually refers to what mathematicians call a tensor field.

Modules

Autograd module

PyTorch uses a method called automatic differentiation. A recorder records what operations have performed, and then it replays it backward to compute the gradients. This method is especially powerful when building neural networks to save time on one epoch by calculating differentiation of the parameters at the forward pass.

Optim module

torch.optim is a module that implements various optimization algorithms used for building neural networks. Most of the commonly used methods are already supported, so there is no need to build them from scratch.

nn module

PyTorch autograd makes it easy to define computational graphs and take gradients, but raw autograd can be a bit too low-level for defining complex neural networks. This is where the nn module can help.

5. THEORY

5.1 Code Design:

A video file containing dashcam footage of a car cruising along the highway is provided for the script laneDetection.py Following a modular approach, the Python script has several functions to perform lane detection.

5.1.1 Image Processing:

readVideo()

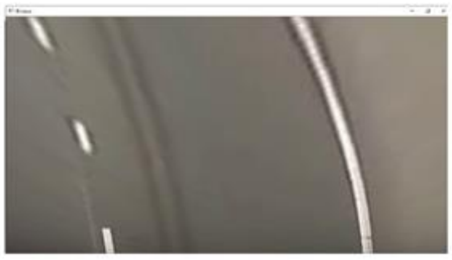
First up is the readVideo() function to access the video file drive.mp4 which is located in the same directory.

processImage()

This function performs some processing techniques to isolate white lane lines and prepare it to be further analyzed by the upcoming functions. Basically, it applies HLS color filtering to filter out whites in the frame, then converts it to grayscale which then is applied thresholding to get rid of unnecessary detections other than lanes, gets blurred and finally edges are extracted with cv2.Canny() function.

perspectiveWarp()

fig 5.1 ROI

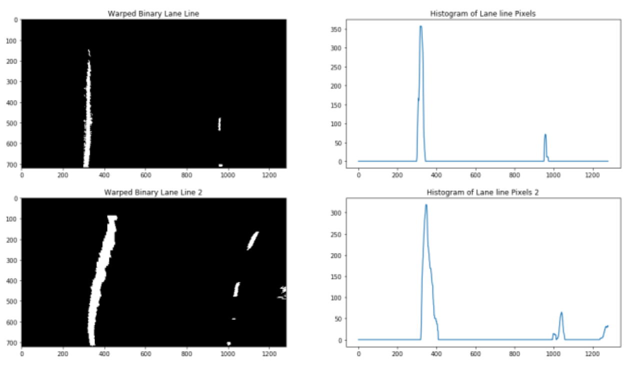


Now that the image that is needed is obtained, a perspective warp is applied. 4 points are placed on the frame such that they surround only the area which lanes are present (like so), then maps it onto another matrix to create a birdseye look at the lanes. This will enable us to work with a much-refined image and help detecting lane curvatures. It should be noted that this operation is subject to change if another video is used. The predefined 4 points are calculated with this particular footage in mind. It should be retuned if another video that has a slightly different angled camera.

5.1.2 Lane Detection, Curve Fitting & Calculations:

plotHistogram()

fig 5.2 Histogram

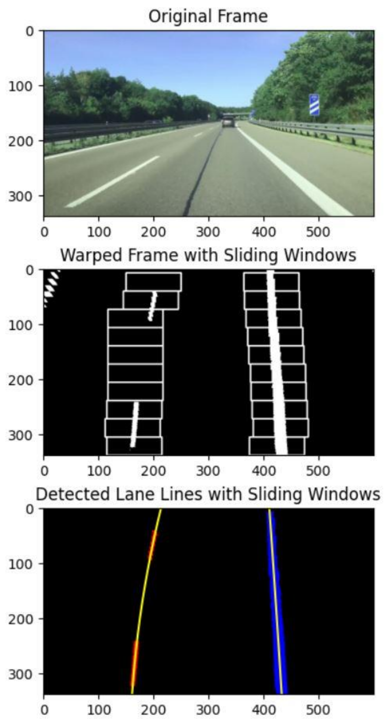


Plotting a histogram for the bottom half of the image is an essential part to obtain the information of where exactly the left and right lanes start. Upon analysing the histogram, one can see there is two distinct peaks where all the white pixels are detected. A very good indicator of where the left and right lanes begin. Since the histogram x coordinate represent the x coordinate of our analysed frame, it means the x coordinates to start searching for the lanes are obtained.

slide_window_search()

A sliding window approach is used to detect lanes and their curvature. It uses information from previous histogram function and puts a box with lane at the center. Then puts another box on top based on the positions of white pixels from the previous box and places itself accordingly all the way to the top of the frame. This way, the information to make some calculations is obtained. Then, a second-degree polynomial fit is performed to have a curve fit in pixel space.

general_search()


Original Frame
Warped Frame with Sliding Windows
Detected Lane Lines with Sliding Windows

After running the slide_window_search() function, this general_search() function is now able to fill up an area around those detected lanes, again applies the second degree polyfit to then draw a yellow line which overlaps the lanes pretty accurately. This line will be used to measure radius of curvature which is essential in predicting steering angles.

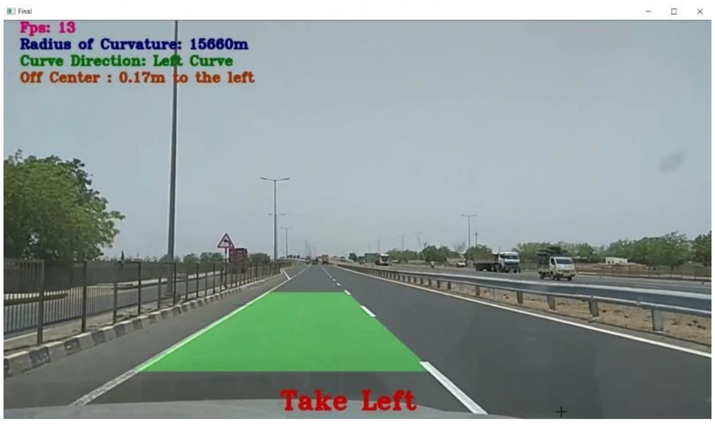
measure_lane_curvature()

With information provided by the previous two functions, np.polyfit() function is used again but with the values multiplied by xm_per_pix and ym_per_pix variables to convert them from pixel space to meter space. xm_per_pix is set as 3.7 / 720 which lane width as 3.7 meters and left & right lane base x coordinates obtained from histogram corresponds to lane width in pixels which turns out to be approximately 720 pixels. Similarly, ym_per_pix is set to 30 / 720 since the frame height is 720.

5.1.3 Visualization and Main Function

draw_lane_lines()

fig 5.3 Lane Calculation



```
Fps: 13
Radius of Curvature: 15660m
Curve Direction: Left Curve
Off Center : 0.17m to the left
Take Left
```

From here on, some methods are applied to visualize the detected lanes and other information to be displayed for the final image. This particular function takes detected lanes and fills the area inside them with a green color. It also visualizes the center of the lane by taking the mean of left_fitx and right_fitx lists and storing them in pts_mean variable, which then is represented by a yellowish color. This variable is also used to calculate the offset of the vehicle to either side or of it is centered in the lane.

offCenter()

offCenter() function uses pts_mean variable to calculate the offset value and show it in meter space.

addText()

Finally by adding text on the final image would complete the process and the information displayed.

main()

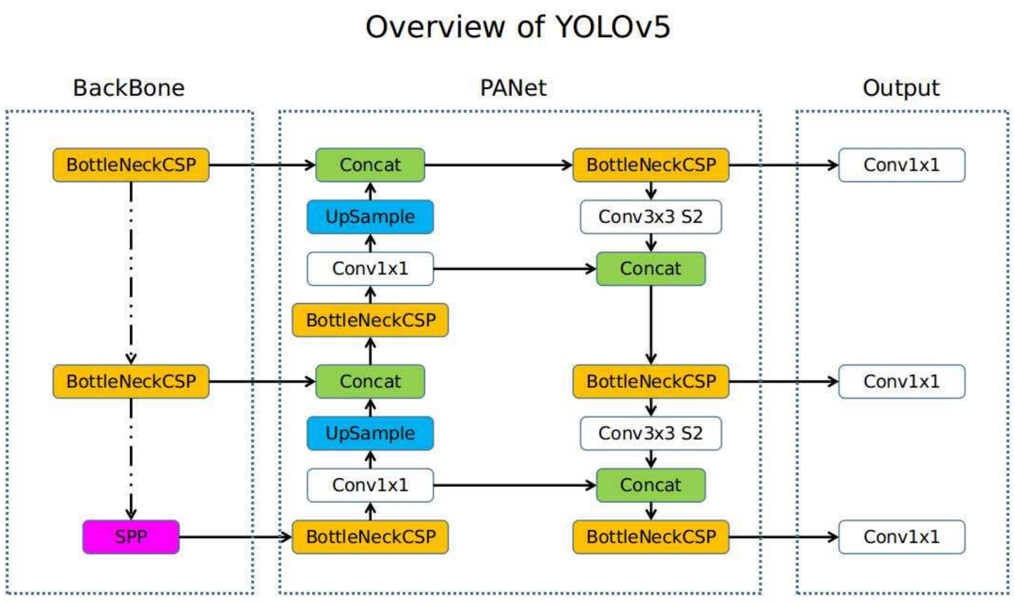
Main function is where all these functions are called in the correct order and contains the loop to play video.

5.2 Object Detection:

YOLO v5s

YOLO is an acronym that stands for You Only Look Once. **Version 5**, which was launched by **Ultralytics** in June 2020 and is now the most advanced object identification algorithm available, is used. It is a novel convolutional neural network (CNN) that detects objects in real-time with great accuracy. This approach uses a single neural network to process the entire picture, then separates it into parts and predicts bounding boxes and probabilities for each component. These bounding boxes are weighted by the expected probability. The method "just looks once" at the image in the sense that it makes predictions after only one forward propagation run through the neural network. It then delivers detected items after non-max suppression (which ensures that the object detection algorithm only identifies each object once).

fig 5.4 YoloV5 Architecture



Its architecture mainly consisted of three parts, namely-

1. Backbone: Model Backbone is mostly used to extract key features from an input image. CSP(Cross Stage Partial Networks) are used as a backbone in YOLO v5 to extract rich in useful characteristics from an input image.

2. Neck: The Model Neck is mostly used to create feature pyramids. Feature pyramids aid models in generalizing successfully when it comes to object scaling. It aids in the identification of the same object in various sizes and scales. Feature pyramids are quite beneficial in assisting models to perform effectively on previously unseen data. Other models, such as FPN, BiFPN, and PANet, use various sorts of feature pyramid approaches.

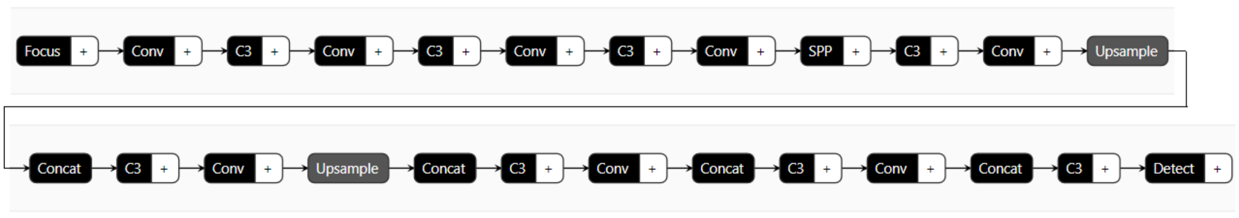
PANet is used as a neck in YOLO v5 to get feature pyramids.

3. Head: The model Head is mostly responsible for the final detection step. It uses anchor boxes to construct final output vectors with class probabilities, objectness scores, and bounding boxes.

Advantages & Disadvantages of Yolo v5s

- It is about 88% smaller than YOLOv4 (27 MB vs 244 MB)
- It is about 180% faster than YOLOv4 (140 FPS vs 50 FPS)
- It is roughly as accurate as YOLOv4 on the same task (0.895 mAP vs 0.892 mAP)
- But the main problem is that for YOLOv5s there is no official paper was released like other YOLO versions. Also, YOLO v5s is still under development and frequent updates from **ultralytics**, is received. developers may update some settings in the future.

YOLOv5s model displayed in Netron



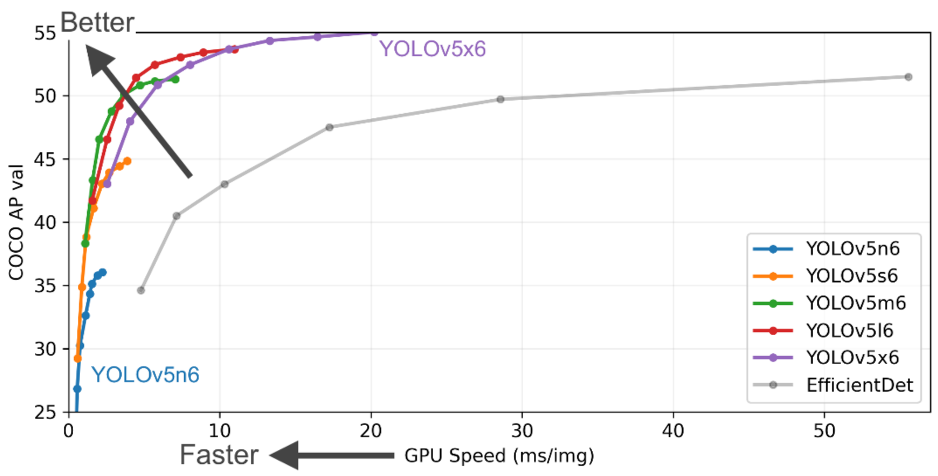


fig 5.5 YoloV5 Metrics

Data Annotation:

An annotation tool is a text or drawing tool that helps you add information to text, an image, a database, or any other piece of content. Here data are annotated using Label-Studio tool.

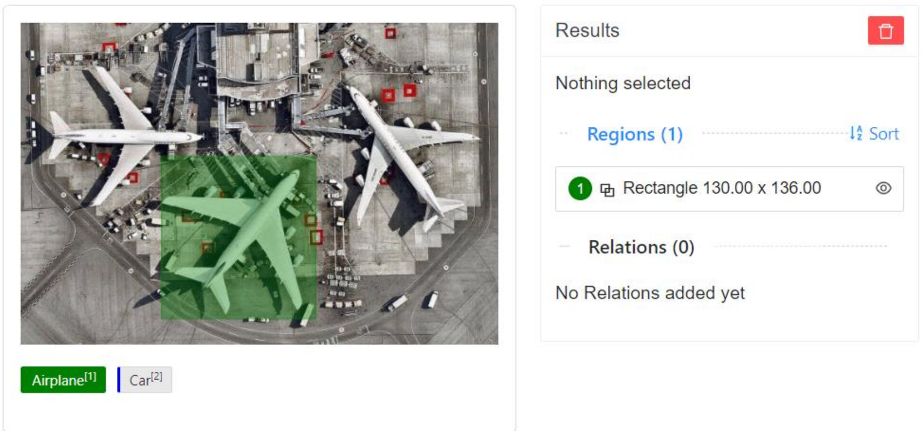
Label Studio is an open source data labelling tool for labelling and exploring multiple types of data. You can perform different types of labelling with many data formats.

1. Install Label Studio:

   pip install label-studio
2. Start Label Studio

   label-studio start
3. Open the Label Studio UI at http://localhost:8080.
4. Sign up with an email address and password that you create.
5. Click **Create** to create a project and start labeling data.
6. Name the project, and if you want, type a description and select a color.
7. Click **Data Import** and upload the data files that you want to use. If you want to use data from a local directory, cloud storage bucket, or database, skip this step for now.
8. Click **Labeling Setup** and choose a template and customize the label names for your use case.
9. Label a region in the data

   - Annotate a section of the data by adding a region.
     Select the label you want to apply to the region. For some configurations, you can skip this step.



   - Click the text, image, audio, or other data object to apply the label to the region. Your changes save automatically.
   - Click **Submit** to submit the completed annotation and move on to the next task.
   - Click **Save** to save your project.

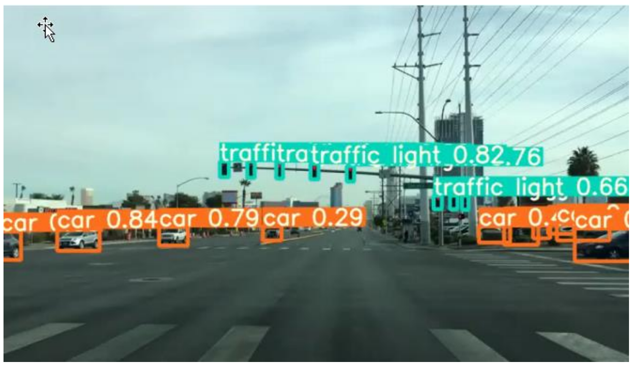
6. WORKING OF PROJECT

The Autonomous Vehicle which has the core system as Nvidia jetson nano, is interfaced with the camera, will stream the real-time video. When the camera attached in front of the vehicle to detects an lane, obstacle, traffic signal or sign boards. Jetson Nano processes the Video data/information and controls/warns the vehicle accordingly.

A Camera is placed in the bonnet of the vehicle, it will capture the real time image of the vehicle front view. Firstly, Video is sent as Frame by frame to Lane detection Algorithm to Compute the camera calibration matrix and distortion coefficients given a set of images. Algorithm Uses color transforms, gradients, etc., to create a thresholded binary image and apply a perspective transform to rectify binary image ("birds-eye view") to detect lane pixels and fit to find the lane boundary. It clearly represents how the way in which the road’s perspective has been changed so as to make computations easier. This helps us in getting the peak of nonzero values of each lane (left and right). Finally, the area around the peak values is added to the list of left and right points into the collected points. The next step is obviously the road-lane detection for which the polynomial regression is used. This can be achieved easily using the Numpy function np.polyfit() in Python. It enables us to approximate the lane/curve by fitting lane points (y) that have been collected with coefficient points (p) into the best approximated lane points (Polyfit) by minimizing the error (E) from the fitted lanes. Hence, by getting right and left points, the functions can be made to approximate the lane structures and to Determine the curvature of the road lane and vehicle position with respect to center. Following this, the prediction of the center of the road by considering the center of the road lane is made and the car is made to steer in that direction. The angle of the servo is calibrated based on how sharp the turn taken to describe the path for self-driving cars and to avoid the risk of getting in another lane. Output visually displayed on the Screen.

Parallelly, each frame feed to Deep Learning algorithm which has revolutionized Computer Vision, and it is the core technology behind capabilities of this self-driving car. Convolutional Neural Networks (CNNs) are at the heart of this deep learning revolution for improving the task of object detection. After receiving the raw input, the feed is passed onto an Al Model. This model is Custom trained using Google Colab Pro. Here using YOLOV5s, an acronym for 'You only look once', is an object detection algorithm that divides images into a grid system. Each cell in the grid is responsible for detecting objects within itself. It is required to add the classes of items that are needed to categorize. Choosing all the classes that are generally visible in the road like all vehicles, road signs, traffic lights, people and animals etc. The output of the YOLO algorithm in its attempt in classifying objects from a sample video feed. It shows the Confidence score and the pixel distance from the camera, which is used for braking of the car and Decision-making system uses its knowledge of objects and other variables to determine what next step to taken by the vehicle like Stop, Turn Left, Turn Right, GO Straight and etc. Path planning system constantly recalibrate the path for vehicle to drive.

7. RESULTS



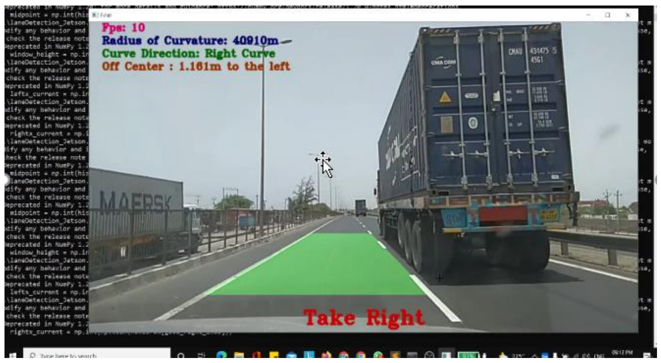

Fps: 10
Radius of Curvature: 40910m
Curve Direction: Right Curve
Off Center : 1.161m to the left
Take Right

fig 7.1 Object Detection

fig7.2 Lane Detection

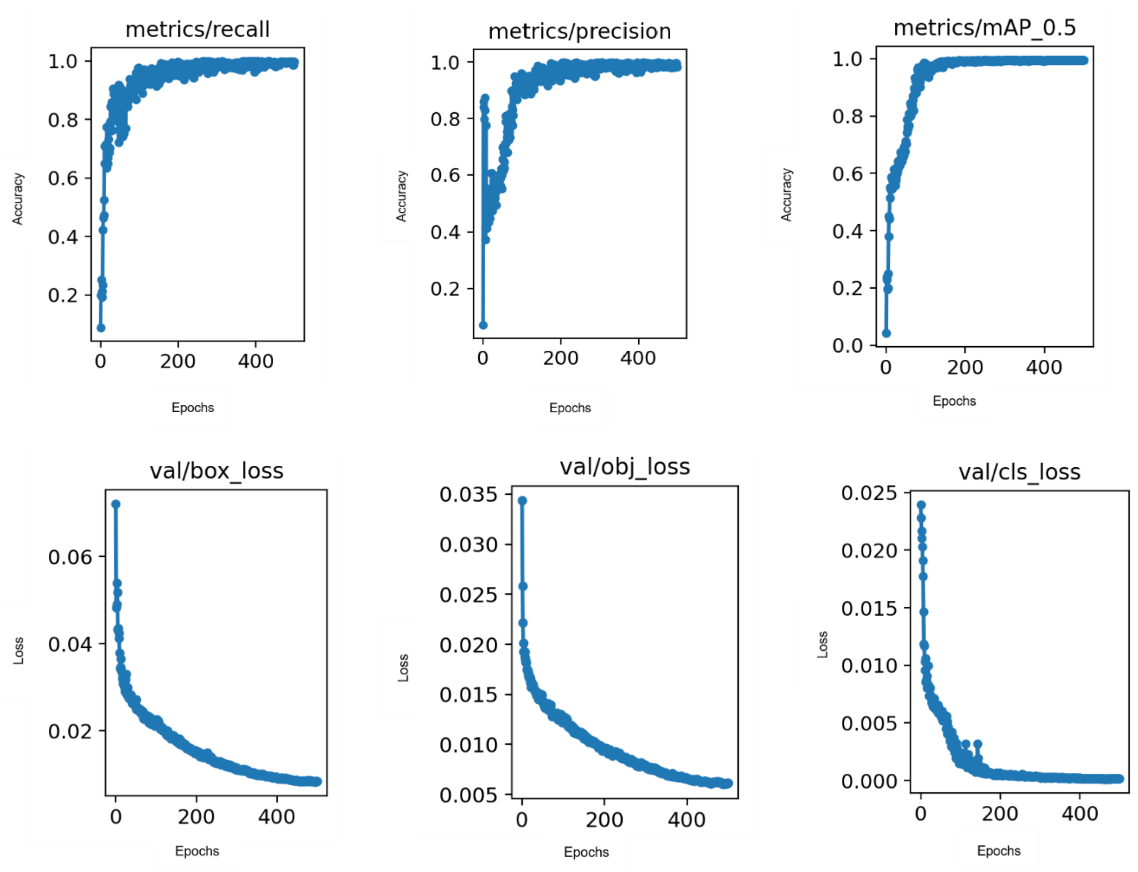


fig 7.3 Loss & Accuracy Metrics of AI Model

Fig 7.1 Displays the Object Detection with bounding boxes that classifies multiple classes. Fig 7.2 displays the lane detection that highlights the lane and displays the coordinates and metrics. Fig 7.3 displays the accuracy metrics, train and validation loss curve.

8. CONCLUSION

The transition from self-driving cars with varying levels of autonomy to fully autonomous vehicles is yet to be made. However, modern AI technologies and machine learning development are making rapid leaps forward in this direction, and that is what’s driving the industry forward. Top automotive brands such as General Motors, Ford, and Tesla are in the final stages of testing their driverless vehicles which means it is are on the verge of seeing a revolutionary change in the way people commute. If the people’s thought hasn’t changed about the self-driving cars being safe, these cars are already safe and are becoming safer. Only if they believe and give a try to technology, they get to enjoy the luxury of computerized driving. If advanced optical metrics like LIDAR are used, then using the information obtained from them, the effectiveness of Autonomous vehicles are significantly improved.

9. APPENDIX

Arduino Code:

```
// Angle = 18 ;  Straight
// Angle = 36 ;  FULL Left
// Angle = 0 ;   FULL Right

// Include the Servo library
#include <Servo.h>
//Motor Driver pins.
#define in1 5           // STEPPER
#define in2 6           // STEPPER
#define in3 10          // Back MOTOR
#define in4 11          // Back MOTOR
#define servoPin 3      //Stepper control pin

int command;            //Int to store app command state.
int Speed = 100;        //Set Speed
int Angle = 18;         //Stepper motor Initial angle
// Create a servo object
Servo Servo;

void setup() {
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  analogWrite(in1, 255);
  analogWrite(in2, 0);

  // Need to attach the servo to the used pin number
   Servo.attach(servoPin);
   Servo.write(18);
  //Set the baud rate to your Bluetooth module.
  Serial.begin(9600);

  // wait for serial port to connect.
  while (!Serial) {;}
}

void forward() {
  Servo.write(Angle);
  analogWrite(in3, Speed);
  analogWrite(in4, 0);
}
```

```
void back() {
  Servo.write(Angle);
  analogWrite(in3, 0);
  analogWrite(in4, Speed);
}

void left() {
  Servo.write(Angle);
  analogWrite(in3, Speed);
  analogWrite(in4, 0);
}

void right() {
  Servo.write(Angle);
  analogWrite(in3, Speed);
  analogWrite(in4, 0);
}

void forwardleft() {
  Servo.write(Angle);
  analogWrite(in3, Speed);
  analogWrite(in4, 0);
}
void forwardright() {
  Servo.write(Angle);
  analogWrite(in3, Speed);
  analogWrite(in4, 0);
}
void backright() {
  Servo.write(Angle);
  analogWrite(in3, 0);
  analogWrite(in4, Speed);
}
void backleft() {
  Servo.write(Angle);
  analogWrite(in3, 0);
  analogWrite(in4, Speed);
}

void stop() {
  Servo.write(Angle);
  analogWrite(in1, 0);
  analogWrite(in2, 0);
  analogWrite(in3, 0);
  analogWrite(in4, 0);
}
```

```
void LeftAngle(){
  if (Angle+6<=36) Angle += 6;
  else Angle = 36;
}

void RightAngle(){
  if (Angle-6>=0) Angle -= 6;
  else Angle = 0;
}

void loop() {
  if (Serial.available() > 0) {
    command = Serial.read();
    Serial.println(char(command));
    //Stop(); //Initialize with motors stoped.

    switch (command) {
      case 'F':
        Angle = 18;
        forward();
        break;
      case 'B':
        Angle = 18;
        back();
        break;
      case 'L':
        LeftAngle();
        left();
        break;
      case 'R':
        RightAngle();
        right();
        break;
      case 'G':
        LeftAngle();
        forwardleft();
        break;
      case 'I':
        RightAngle();
        forwardright();
        break;
      case 'H':
        LeftAngle();
        backleft();
        break;
      case 'J':
        RightAngle();
        backright();
        break;
      case 'S':
        Angle = 18;
        stop();
        break;
      default:
        //forward();
        break;
    }
  }
}
```

AI Model Code:

```
import argparse
import os
import sys
from pathlib import Path

import cv2
import torch
import torch.backends.cudnn as cudnn

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative

from models.common import DetectMultiBackend
from utils.datasets import IMG_FORMATS, VID_FORMATS, LoadImages,
LoadStreams
from utils.general import (LOGGER, check_file, check_img_size,
check_imshow, check_requirements, colorstr,
                           increment_path, non_max_suppression,
print_args, scale_coords, strip_optimizer, xyxy2xywh)
from utils.plots import Annotator, colors, save_one_box
from utils.torch_utils import select_device, time_sync


@torch.no_grad()
def run(weights=ROOT / 'yolov5s.pt',  # model.pt path(s)
        source=ROOT / 'data/images',  # file/dir/URL/glob, 0 for webcam
        data=ROOT / 'data/coco128.yaml',  # dataset.yaml path
        imgsz=(640, 640),  # inference size (height, width)
        conf_thres=0.25,  # confidence threshold
        iou_thres=0.45,  # NMS IOU threshold
        max_det=1000,  # maximum detections per image
        device='',  # cuda device, i.e. 0 or 0,1,2,3 or cpu
        view_img=False,  # show results
        save_txt=False,  # save results to *.txt
        save_conf=False,  # save confidences in --save-txt labels
        save_crop=False,  # save cropped prediction boxes
        nosave=False,  # do not save images/videos
        classes=None,  # filter by class: --class 0, or --class 0 2 3
        agnostic_nms=False,  # class-agnostic NMS
        augment=False,  # augmented inference
        visualize=False,  # visualize features
        update=False,  # update all models
        project=ROOT / 'runs/detect',  # save results to project/name
        name='exp',  # save results to project/name
        exist_ok=False,  # existing project/name ok, do not increment
        line_thickness=3,  # bounding box thickness (pixels)
        hide_labels=False,  # hide labels
        hide_conf=False,  # hide confidences
        half=False,  # use FP16 half-precision inference
        dnn=False,  # use OpenCV DNN for ONNX inference
        ):
    source = str(source)
    save_img = not nosave and not source.endswith('.txt')  # save
inference images
    is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
    is_url = source.lower().startswith(('rtsp://', 'rtmp://',
'http://', 'https://'))
    webcam = source.isnumeric() or source.endswith('.txt') or (is_url
and not is_file)
    if is_url and is_file:
        source = check_file(source)  # download

    # Directories
    save_dir = increment_path(Path(project) / name,
exist_ok=exist_ok)  # increment run
    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True,
exist_ok=True)  # make dir

    # Load model
    device = select_device(device)
    model = DetectMultiBackend(weights, device=device, dnn=dnn,
data=data, fp16=half)
    stride, names, pt = model.stride, model.names, model.pt
    imgsz = check_img_size(imgsz, s=stride)  # check image size

    # Dataloader
    if webcam:
        view_img = check_imshow()
        cudnn.benchmark = True  # set True to speed up constant image
size inference
        dataset = LoadStreams(source, img_size=imgsz, stride=stride,
auto=pt)
        bs = len(dataset)  # batch_size
    else:
        dataset = LoadImages(source, img_size=imgsz, stride=stride,
auto=pt)
        bs = 1  # batch_size
    vid_path, vid_writer = [None] * bs, [None] * bs

    # Run inference
    model.warmup(imgsz=(1 if pt else bs, 3, *imgsz))  # warmup
    dt, seen = [0.0, 0.0, 0.0], 0
    for path, im, im0s, vid_cap, s in dataset:
        t1 = time_sync()
        im = torch.from_numpy(im).to(device)
        im = im.half() if model.fp16 else im.float()  # uint8 to
fp16/32
        im /= 255  # 0 - 255 to 0.0 - 1.0
        if len(im.shape) == 3:
            im = im[None]  # expand for batch dim
        t2 = time_sync()
        dt[0] += t2 - t1

        # Inference
        visualize = increment_path(save_dir / Path(path).stem,
mkdir=True) if visualize else False
        pred = model(im, augment=augment, visualize=visualize)
        t3 = time_sync()
        dt[1] += t3 - t2

        # NMS
        pred = non_max_suppression(pred, conf_thres, iou_thres,
classes, agnostic_nms, max_det=max_det)
        dt[2] += time_sync() - t3

        # Second-stage classifier (optional)
        # pred = utils.general.apply_classifier(pred, classifier_model,
im, im0s)

        # Process predictions
        for i, det in enumerate(pred):  # per image
            seen += 1
            if webcam:  # batch_size >= 1
                p, im0, frame = path[i], im0s[i].copy(), dataset.count
                s += f'{i}: '
            else:
                p, im0, frame = path, im0s.copy(), getattr(dataset,
'frame', 0)

            p = Path(p)  # to Path
            save_path = str(save_dir / p.name)  # im.jpg
            txt_path = str(save_dir / 'labels' / p.stem) + ('' if
dataset.mode == 'image' else f'_{frame}')  # im.txt
```

```
            s += '%gx%g ' % im.shape[2:]  # print string
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization
gain whwh
            imc = im0.copy() if save_crop else im0  # for save_crop
            annotator = Annotator(im0, line_width=line_thickness,
example=str(names))
            if len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_coords(im.shape[2:], det[:, :4],
im0.shape).round()

                # Print results
                for c in det[:, -1].unique():
                    n = (det[:, -1] == c).sum()  # detections per class
                    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add
to string

                # Write results
                for *xyxy, conf, cls in reversed(det):
                    if save_txt:  # Write to file
                        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1,
4)) / gn).view(-1).tolist()  # normalized xywh
                        line = (cls, *xywh, conf) if save_conf else
(cls, *xywh)  # label format
                        with open(txt_path + '.txt', 'a') as f:
                            f.write(('%g ' * len(line)).rstrip() % line
+ '\n')

                    if save_img or save_crop or view_img:  # Add bbox
to image
                        c = int(cls)  # integer class
                        label = None if hide_labels else (names[c] if
hide_conf else f'{names[c]} {conf:.2f}')
                        annotator.box_label(xyxy, label,
color=colors(c, True))
                        if save_crop:
                            save_one_box(xyxy, imc, file=save_dir /
'crops' / names[c] / f'{p.stem}.jpg', BGR=True)

            # Stream results
            im0 = annotator.result()
            if view_img:
                cv2.imshow(str(p), im0)
                cv2.waitKey(1)  # 1 millisecond

            # Save results (image with detections)
            if save_img:
                if dataset.mode == 'image':
                    cv2.imwrite(save_path, im0)
                else:  # 'video' or 'stream'
                    if vid_path[i] != save_path:  # new video
                        vid_path[i] = save_path
                        if isinstance(vid_writer[i], cv2.VideoWriter):
                            vid_writer[i].release()  # release previous
video writer
                        if vid_cap:  # video
                            fps = vid_cap.get(cv2.CAP_PROP_FPS)
                            w =
int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                            h =
int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                        else:  # stream
                            fps, w, h = 30, im0.shape[1], im0.shape[0]
                        save_path =
str(Path(save_path).with_suffix('.mp4'))  # force *.mp4 suffix on
results videos
                        vid_writer[i] = cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
                    vid_writer[i].write(im0)

        # Print time (inference-only)
        LOGGER.info(f'{s}Done. ({t3 - t2:.3f}s)')

    # Print results
    t = tuple(x / seen * 1E3 for x in dt)  # speeds per image
    LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference, %.1fms
NMS per image at shape {(1, 3, *imgsz)}' % t)
    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved
to {save_dir / 'labels'}" if save_txt else ''
        LOGGER.info(f"Results saved to {colorstr('bold',
save_dir)}{s}")
    if update:
        strip_optimizer(weights)  # update model (to fix
SourceChangeWarning)


def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT
/ 'yolov5s.pt', help='model path(s)')
```

```
    parser.add_argument('--source', type=str, default=ROOT / 
'data/images', help='file/dir/URL/glob, 0 for webcam')
    parser.add_argument('--data', type=str, default=ROOT / 
'data/coco128.yaml', help='(optional) dataset.yaml path')
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', 
type=int, default=[640], help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.25, 
help='confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, 
help='NMS IoU threshold')
    parser.add_argument('--max-det', type=int, default=1000, 
help='maximum detections per image')
    parser.add_argument('--device', default='', help='cuda device, i.e. 
0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='show 
results')
    parser.add_argument('--save-txt', action='store_true', help='save 
results to *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save 
confidences in --save-txt labels')
    parser.add_argument('--save-crop', action='store_true', help='save 
cropped prediction boxes')
    parser.add_argument('--nosave', action='store_true', help='do not 
save images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter 
by class: --classes 0, or --classes 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', 
help='class-agnostic NMS')
    parser.add_argument('--augment', action='store_true', 
help='augmented inference')
    parser.add_argument('--visualize', action='store_true', 
help='visualize features')
    parser.add_argument('--update', action='store_true', help='update 
all models')
    parser.add_argument('--project', default=ROOT / 'runs/detect', 
help='save results to project/name')
    parser.add_argument('--name', default='exp', help='save results to 
project/name')
    parser.add_argument('--exist-ok', action='store_true', 
help='existing project/name ok, do not increment')
    parser.add_argument('--line-thickness', default=3, type=int, 
help='bounding box thickness (pixels)')
    parser.add_argument('--hide-labels', default=False, 
action='store_true', help='hide labels')
    parser.add_argument('--hide-conf', default=False, 
action='store_true', help='hide confidences')
```

```
    parser.add_argument('--half', action='store_true', help='use FP16 
half-precision inference')
    parser.add_argument('--dnn', action='store_true', help='use OpenCV 
DNN for ONNX inference')
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1  # expand
    print_args(FILE.stem, opt)
    return opt


def main(opt):
    check_requirements(exclude=('tensorboard', 'thop'))
    run(**vars(opt))


if __name__ == "__main__":
    opt = parse_opt()
    main(opt)
```

Jetson Code:

```
# IMPORT NECESSARY LIBRARIES
import cv2
import numpy as np
import os
from scipy import optimize
from matplotlib import pyplot as plt, cm, colors
import serial
import time

# Defining variables to hold meter-to-pixel conversion
ym_per_pix = 30 / 720
# Standard lane width is 3.7 meters divided by lane width in pixels 
which is
# calculated to be approximately 720 pixels not to be confused with 
frame height
xm_per_pix = 3.7 / 720

#define Variable to calculate Frame Rate
new_frame,pre_frame = 0,0

# Get path to the current working directory
CWD_PATH = os.getcwd()

ser = serial.Serial('/dev/ttyACM0', baudrate=9600, timeout=10)


def readVideo():

    # Read input video from current working directory
    #inpImage = cv2.VideoCapture(os.path.join(CWD_PATH, 
'indian_road.mp4'))
    inpImage  = cv2.VideoCapture(0)

    return inpImage

def processImage(inpImage):

    # Apply HLS color filtering to filter out white lane lines
    hls = cv2.cvtColor(inpImage, cv2.COLOR_BGR2HLS)
    lower_white = np.array([0, 160, 10])
    upper_white = np.array([255, 255, 255])
    mask = cv2.inRange(inpImage, lower_white, upper_white)
    hls_result = cv2.bitwise_and(inpImage, inpImage, mask = mask)
    # Convert image to grayscale, apply threshold, blur & extract edges
    gray = cv2.cvtColor(hls_result, cv2.COLOR_BGR2GRAY)
    ret, thresh = cv2.threshold(gray, 160, 255, cv2.THRESH_BINARY)
    blur = cv2.GaussianBlur(thresh,(3, 3), 0)
    canny = cv2.Canny(blur, 40, 60)

    # Display the processed images
##    cv2.imshow("Image", inpImage)
##    cv2.imshow("HLS Filtered", hls_result)
##    cv2.imshow("Grayscale", gray)
##    cv2.imshow("Thresholded", thresh)
##    cv2.imshow("Blurred", blur)
##    cv2.imshow("Canny Edges", canny)

    return image, hls_result, gray, thresh, blur, canny


def perspectiveWarp(inpImage):

    # Get image size
    img_size = (inpImage.shape[1], inpImage.shape[0])

    # Perspective points to be warped
    #[[Top_left],[Top_right],[Bottom_left],[Bottom_right]]
    src = np.float32([[270,600],[1050,600],[140,920],[1170,920]])
    #src = np.float32([[457, 490],[650, 490],[108, 625],[850, 625]])

    # Window to be shown
    #dst = np.float32([[200, 0],[1200, 0],[200, 710],[1200, 710]])
    dst = np.float32([[200, 0],[1200, 0],[200, 710],[1200, 710]])

    # Matrix to warp the image for birdseye window
    matrix = cv2.getPerspectiveTransform(src, dst)
    # Inverse matrix to unwarp the image for final window
    minv = cv2.getPerspectiveTransform(dst, src)
    birdseye = cv2.warpPerspective(inpImage, matrix, img_size)

    # Get the birdseye window dimensions
    height, width = birdseye.shape[:2]

    # Divide the birdseye view into 2 halves to separate left & right
lanes
    birdseyeLeft  = birdseye[0:height, 0:width // 2]
    birdseyeRight = birdseye[0:height, width // 2:width]

    # Display birdseye view image
    #cv2.imshow("Birdseye" , birdseye)
    # cv2.imshow("Birdseye Left" , birdseyeLeft)
    # cv2.imshow("Birdseye Right", birdseyeRight)

    return birdseye, birdseyeLeft, birdseyeRight, minv

def plotHistogram(inpImage):

    histogram = np.sum(inpImage[inpImage.shape[0] // 2:, :], axis = 0)

    midpoint = np.int(histogram.shape[0] / 2)
    leftxBase = np.argmax(histogram[:midpoint])
    rightxBase = np.argmax(histogram[midpoint:]) + midpoint

    plt.xlabel("Image X Coordinates")
    plt.ylabel("Number of White Pixels")

    # Return histogram and x-coordinates of left & right lanes to
calculate
    # lane width in pixels
    return histogram, leftxBase, rightxBase

def slide_window_search(binary_warped, histogram):

    # Find the start of left and right lane lines using histogram info
    out_img = np.dstack((binary_warped, binary_warped, binary_warped))
* 255
    midpoint = np.int(histogram.shape[0] / 2)
    leftx_base = np.argmax(histogram[:midpoint])
    rightx_base = np.argmax(histogram[midpoint:]) + midpoint

    # A total of 9 windows will be used
    nwindows = 9
    window_height = np.int(binary_warped.shape[0] / nwindows)
    nonzero = binary_warped.nonzero()
    nonzeroy = np.array(nonzero[0])
    nonzerox = np.array(nonzero[1])
    leftx_current = leftx_base
    rightx_current = rightx_base
    margin = 100
    minpix = 50
    left_lane_inds = []
    right_lane_inds = []

    #### START - Loop to iterate through windows and search for lane
lines #####
```

```
    for window in range(nwindows):
        win_y_low = binary_warped.shape[0] - (window + 1) *
window_height
        win_y_high = binary_warped.shape[0] - window * window_height
        win_xleft_low = leftx_current - margin
        win_xleft_high = leftx_current + margin
        win_xright_low = rightx_current - margin
        win_xright_high = rightx_current + margin
        cv2.rectangle(out_img, (win_xleft_low, win_y_low),
(win_xleft_high, win_y_high),
        (0,255,0), 2)
        cv2.rectangle(out_img, (win_xright_low,win_y_low),
(win_xright_high,win_y_high),
        (0,255,0), 2)
        good_left_inds = ((nonzeroy >= win_y_low) & (nonzeroy <
win_y_high) &
        (nonzerox >= win_xleft_low) &  (nonzerox <
win_xleft_high)).nonzero()[0]
        good_right_inds = ((nonzeroy >= win_y_low) & (nonzeroy <
win_y_high) &
        (nonzerox >= win_xright_low) &  (nonzerox <
win_xright_high)).nonzero()[0]
        left_lane_inds.append(good_left_inds)
        right_lane_inds.append(good_right_inds)
        if len(good_left_inds) > minpix:
            leftx_current = np.int(np.mean(nonzerox[good_left_inds]))
        if len(good_right_inds) > minpix:
            rightx_current = np.int(np.mean(nonzerox[good_right_inds]))
    #### END - Loop to iterate through windows and search for lane
lines #######

    left_lane_inds = np.concatenate(left_lane_inds)
    right_lane_inds = np.concatenate(right_lane_inds)

    leftx = nonzerox[left_lane_inds]
    lefty = nonzeroy[left_lane_inds]
    rightx = nonzerox[right_lane_inds]
    righty = nonzeroy[right_lane_inds]

    # Apply 2nd degree polynomial fit to fit curves
    left_fit = np.polyfit(lefty, leftx, 2)
    right_fit = np.polyfit(righty, rightx, 2)


    ploty = np.linspace(0, binary_warped.shape[0]-1,
binary_warped.shape[0])
```

```
    left_fitx = left_fit[0] * ploty**2 + left_fit[1] * ploty +
left_fit[2]
    right_fitx = right_fit[0] * ploty**2 + right_fit[1] * ploty +
right_fit[2]

    ltx = np.trunc(left_fitx)
    rtx = np.trunc(right_fitx)
    plt.plot(right_fitx)
    # plt.show()

    out_img[nonzeroy[left_lane_inds], nonzerox[left_lane_inds]] = [255,
0, 0]
    out_img[nonzeroy[right_lane_inds], nonzerox[right_lane_inds]] = [0,
0, 255]

    # plt.imshow(out_img)
    plt.plot(left_fitx,  ploty, color = 'yellow')
    plt.plot(right_fitx, ploty, color = 'yellow')
    plt.xlim(0, 1280)
    plt.ylim(720, 0)

    return ploty, left_fit, right_fit, ltx, rtx

def general_search(binary_warped, left_fit, right_fit):

    nonzero = binary_warped.nonzero()
    nonzeroy = np.array(nonzero[0])
    nonzerox = np.array(nonzero[1])
    margin = 100
    left_lane_inds = ((nonzerox > (left_fit[0]*(nonzeroy**2) +
left_fit[1]*nonzeroy +
    left_fit[2] - margin)) & (nonzerox < (left_fit[0]*(nonzeroy**2) +
    left_fit[1]*nonzeroy + left_fit[2] + margin)))

    right_lane_inds = ((nonzerox > (right_fit[0]*(nonzeroy**2) +
right_fit[1]*nonzeroy +
    right_fit[2] - margin)) & (nonzerox < (right_fit[0]*(nonzeroy**2) +
    right_fit[1]*nonzeroy + right_fit[2] + margin)))

    leftx = nonzerox[left_lane_inds]
    lefty = nonzeroy[left_lane_inds]
    rightx = nonzerox[right_lane_inds]
    righty = nonzeroy[right_lane_inds]
    left_fit = np.polyfit(lefty, leftx, 2)
    right_fit = np.polyfit(righty, rightx, 2)
    ploty = np.linspace(0, binary_warped.shape[0]-1,
binary_warped.shape[0])
    left_fitx = left_fit[0]*ploty**2 + left_fit[1]*ploty + left_fit[2]
    right_fitx = right_fit[0]*ploty**2 + right_fit[1]*ploty +
right_fit[2]


    ## VISUALIZATION
############################################################

    out_img = np.dstack((binary_warped, binary_warped,
binary_warped))*255
    window_img = np.zeros_like(out_img)
    out_img[nonzeroy[left_lane_inds], nonzerox[left_lane_inds]] = [255,
0, 0]
    out_img[nonzeroy[right_lane_inds], nonzerox[right_lane_inds]] = [0,
0, 255]

    left_line_window1 = np.array([np.transpose(np.vstack([left_fitx-
margin, ploty]))])
    left_line_window2 =
np.array([np.flipud(np.transpose(np.vstack([left_fitx+margin,
                                  ploty])))])
    left_line_pts = np.hstack((left_line_window1, left_line_window2))
    right_line_window1 = np.array([np.transpose(np.vstack([right_fitx-
margin, ploty]))])
    right_line_window2 =
np.array([np.flipud(np.transpose(np.vstack([right_fitx+margin,
ploty])))])
    right_line_pts = np.hstack((right_line_window1,
right_line_window2))

    cv2.fillPoly(window_img, np.int_([left_line_pts]), (0, 255, 0))
    cv2.fillPoly(window_img, np.int_([right_line_pts]), (0, 255, 0))
    result = cv2.addWeighted(out_img, 1, window_img, 0.3, 0)

    # plt.imshow(result)
    plt.plot(left_fitx,  ploty, color = 'yellow')
    plt.plot(right_fitx, ploty, color = 'yellow')
    plt.xlim(0, 1280)
    plt.ylim(720, 0)

    ret = {}
    ret['leftx'] = leftx
    ret['rightx'] = rightx
    ret['left_fitx'] = left_fitx
    ret['right_fitx'] = right_fitx
    ret['ploty'] = ploty

    return ret


def measure_lane_curvature(ploty, leftx, rightx):

    leftx = leftx[::-1]  # Reverse to match top-to-bottom in y
    rightx = rightx[::-1]  # Reverse to match top-to-bottom in y

    # Choose the maximum y-value, corresponding to the bottom of the
image
    y_eval = np.max(ploty)

    # Fit new polynomials to x, y in world space
    left_fit_cr = np.polyfit(ploty*ym_per_pix, leftx*xm_per_pix, 2)
    right_fit_cr = np.polyfit(ploty*ym_per_pix, rightx*xm_per_pix, 2)

    # Calculate the new radii of curvature
    left_curverad  = ((1 + (2*left_fit_cr[0]*y_eval*ym_per_pix +
left_fit_cr[1])**2)**1.5) / np.absolute(2*left_fit_cr[0])
    right_curverad = ((1 + (2*right_fit_cr[0]*y_eval*ym_per_pix +
right_fit_cr[1])**2)**1.5) / np.absolute(2*right_fit_cr[0])
    # Now our radius of curvature is in meters
    # print(left_curverad, 'm', right_curverad, 'm')

    # Decide if it is a left or a right curve
    if leftx[0] - leftx[-1] > 60:
        curve_direction = 'Left Curve'
    elif leftx[-1] - leftx[0] > 60:
        curve_direction = 'Right Curve'
    else:
        curve_direction = 'Straight'

    return (left_curverad + right_curverad) / 2.0, curve_direction


def draw_lane_lines(original_image, warped_image, Minv, draw_info):

    leftx = draw_info['leftx']
    rightx = draw_info['rightx']
    left_fitx = draw_info['left_fitx']
    right_fitx = draw_info['right_fitx']
    ploty = draw_info['ploty']
    warp_zero = np.zeros_like(warped_image).astype(np.uint8)
    color_warp = np.dstack((warp_zero, warp_zero, warp_zero))

    pts_left = np.array([np.transpose(np.vstack([left_fitx, ploty]))])
    pts_right = np.array([np.flipud(np.transpose(np.vstack([right_fitx,
ploty])))])
    pts = np.hstack((pts_left, pts_right))

    mean_x = np.mean((left_fitx, right_fitx), axis=0)
    pts_mean = np.array([np.flipud(np.transpose(np.vstack([mean_x,
ploty])))])

    cv2.fillPoly(color_warp, np.int_([pts]), (0, 255, 0))
    cv2.fillPoly(color_warp, np.int_([pts_mean]), (0, 255, 255))

    newwarp = cv2.warpPerspective(color_warp, Minv,
(original_image.shape[1], original_image.shape[0]))
    result = cv2.addWeighted(original_image, 1, newwarp, 0.3, 0)

    return pts_mean, result

def offCenter(meanPts, inpFrame):

    # Calculating deviation in meters
    mpts = meanPts[-1][-1][-2].astype(int)
    pixelDeviation = inpFrame.shape[1] / 2 - abs(mpts)
    deviation = pixelDeviation * xm_per_pix
    direction = "left" if deviation < 0 else "right"

    return deviation, direction

def addText(img, radius, direction, deviation, devDirection):

    # Add the radius and center position to the image
    font = cv2.FONT_HERSHEY_TRIPLEX

    if (direction != 'Straight'):
        text = 'Radius of Curvature: ' + '{:04.0f}'.format(radius) +
'm'
        text1 = 'Curve Direction: ' + (direction)

    else:
        text = 'Radius of Curvature : ' + 'N/A'
        text1 = 'Curve Direction : ' + (direction)
    cv2.putText(img, text , (30,50), font, 0.8, (100,0, 0), 2,
cv2.LINE_AA)
    cv2.putText(img, text1, (30,80), font, 0.8, (0,100, 0), 2,
cv2.LINE_AA)

    # Deviation
    deviation_text = 'Off Center : ' + str(round(abs(deviation), 3)) +
'm' + ' to the ' + devDirection
    cv2.putText(img, deviation_text, (30, 110),
cv2.FONT_HERSHEY_TRIPLEX, 0.8, (0,50, 150), 2, cv2.LINE_AA)

    if direction == 'Straight':
        cv2.putText(img, "GO Straight", (500, 900), font, 1.5, (0, 0,
200), 3, cv2.LINE_AA)
        ser.write(bytes('F\n','utf-8'))
    elif direction == 'Left Curve':
        cv2.putText(img, "Take Left", (500, 900), font, 1.5, (0, 0,
200), 3, cv2.LINE_AA)
        ser.write(bytes('G\n','utf-8'))
    elif direction == 'Right Curve':
        cv2.putText(img, "Take Right", (500, 900), font, 1.5, (0, 0,
200), 3, cv2.LINE_AA)
        ser.write(bytes('I\n','utf-8'))
    return img

#------------------------------------------------------------------------
---------------------------------------------

#---------------------------------------------Main---------------------
---------------------------------------------

# Read the input image
image = readVideo()

while True:

    _, frame = image.read()

    start = time.time()
    # Apply perspective warping by calling the "perspectiveWarp()"
function
    # Then assign it to the variable called (birdView)
    # Provide this function with:
    # 1- an image to apply perspective warping (frame)
    birdView, birdViewL, birdViewR, minverse = perspectiveWarp(frame)
    # Apply image processing by calling the "processImage()" function
    # Then assign their respective variables (img, hls, grayscale,
thresh, blur, canny)
    # Provide this function with:
    # 1- an already perspective warped image to process (birdView)
    img, hls, grayscale, thresh, blur, canny = processImage(birdView)
    imgL, hlsL, grayscaleL, threshL, blurL, cannyL =
processImage(birdViewL)
    imgR, hlsR, grayscaleR, threshR, blurR, cannyR =
processImage(birdViewR)


    # Plot and display the histogram by calling the "get_histogram()"
function
    # Provide this function with:
    # 1- an image to calculate histogram on (thresh)
    hist, leftBase, rightBase = plotHistogram(thresh)
    # print(rightBase - leftBase)
    plt.plot(hist)
    # plt.show()


    ploty, left_fit, right_fit, left_fitx, right_fitx =
slide_window_search(thresh, hist)
    plt.plot(left_fit)
    # plt.show()


    draw_info = general_search(thresh, left_fit, right_fit)
    # plt.show()


    curveRad, curveDir = measure_lane_curvature(ploty, left_fitx,
right_fitx)


    # Filling the area of detected lanes with green
    meanPts, result = draw_lane_lines(frame, thresh, minverse,
draw_info)


    deviation, directionDev = offCenter(meanPts, frame)


    # Adding text to our final image
    finalImg = addText(result, curveRad, curveDir, deviation,
directionDev)
```

```
    #Adding Frame Rate our final image
    new_frame = time.time()
    fps = 1/(new_frame-pre_frame)
    pre_frame = new_frame
    cv2.putText(finalImg, "Fps: "+str(int(fps)), (30,20),
cv2.FONT_HERSHEY_TRIPLEX, 0.8, (100,0, 200), 2, cv2.LINE_AA)

    # Displaying final image
    cv2.imshow("Final", finalImg)
    cv2.moveWindow("Final", 100, 0)

    end = time.time()

    #print("FPS : ",round(fps,5),"Time : ",end-start)


    # Wait for the ENTER key to be pressed to stop playback
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

ser.write(bytes('S\n','utf-8')) #Stop_Vehicle
image.release()
cv2.destroyAllWindows()

#######################################################################
#####################END_CODE_FUNCTION#################################
```

9. REFERENCE

[1] Automated Vehicles Safety, Abul Hasan Fahad, in Solving Urban Infrastructure Problems Using Smart City Technologies

[2] Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments, Raphael E. Stern, Shumo Cui

[3] Autonomous vehicles: The future of automobiles, M V Rajasekhar; Anil Kumar Jaswal

[4] AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles, Shital Shah, Debadeepta Dey, Chris Lovett, Ashish Kapoor