URKUND

Urkund Analysis Result

Analysed Document: palgarithmcheaked.docx (D44155887)
Submitted: 11/18/2018 3:23:00 PM
Submitted By: selvakumar.r@kluniversity.in
Significance: 0 %

Sources included in the report:

Instances where selected sources appear:

0

ABSTRACT

Autonomous vehicles are very famous now a day's. In this project we develop a autonomous vehicle prototype. We use Machine learning concepts to achieve autonomous driving. We initially train the vehicle manually through remote access using internet. During this training we obtain data from sensors regarding object distance around the vehicle at every instance of time and current direction of the vehicle. Later we feed this data into machine learning algorithms and develop a classifier which predicts the directions for new sensor data using previous experience. This project also analyzes the effect of different algorithms on the vehicle and accuracy comparisons between those algorithms. In this project we also developed in-depth procedure to build an autonomous vehicle prototype from scratch. Both hardware and software architectures are developed and mentioned in detail.

CHAPTER 1 INTRODUCTION

1.1 Autonomous vehicles:- Autonomous vehicle are known concept today in the industry. Autonomous vehicles drive autonomously without any assistance from the driver. This technology saves a lot of time to the driver. An average humans speed 1 year of their life span in driving. By saving this time a lot of productive work can be produced. Autonomous vehicles are combination of field of study of navigation, data science, and embedded systems. Historically autonomous vehicles are derived from auto pilot implemented to flight technologies. The concept of autonomous vehicle is first presented by Google in Google I/O 2013, later which was developed into Google self-driving vehicle. Now-a-days companies like Tesla, Uber, Volvo are investing millions of dollars in research and development of self-driving vehicles. Autonomous vehicles are classified into 6 levels based on Autonomous level by SAE International. SAE LEVEL NAME Description 0 No Automation The full-time performance by the human driver of all aspects of the dynamic driving task, even when "enhanced by warning or intervention systems" 1 Driver Assistance The driving mode-specific execution by a driver assistance system of "either steering or acceleration/deceleration" 2 Partial Automation The driving mode-specific execution by one or more driver assistance systems of both steering and acceleration or deceleration 3 Conditional Automation

The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task with the expectation that the human driver will respond appropriately to a request to intervene 4 High Automation even if a human driver does not respond appropriately to a request to intervene 5 Full Automation under all roadway and environmental conditions that can be managed by a human driver Table 1.1.1 Types of Automation Autonomous vehicles are needed to be trained by a driver in real-time environment. So most of the companies build prototypes of their research and train them around the globe the training time and distance data are as follows Maker 2016

Distance between disengagements Distance Delphi Automotive Systems 14.9 miles (24.0 km) 2,658 miles (4,278 km) Tesla 2.9 miles (4.7 km) 550 miles (890 km) Ford 196.6 miles (316.4 km) 590 miles (950 km) Nissan 263.3 miles (423.7 km) 6,056 miles (9,746 km) Waymo 5,127.9 miles (8,252.6 km) 635,868 miles (1,023,330 km) BMW 638 miles (1,027 km) 638 miles (1,027 km) Mercedes Benz 2 miles (3.2 km) 673 miles (1,083 km) General Motors 54.7 miles (88.0 km) 8,156 miles (13,126 km) Bosch 0.68 miles (1.09 km) 983 les (1,582 km)

Table 1.1.2 Training details of different companies 1.2 Process Flow:- There are many fields involved in developing an autonomous vehicle. We follow a practical approach where we develop a prototype of a autonomous vehicle. We incorporate all the fields involved in the development. The most significant field of study are mentioned below 1. Data Science a. Machine learning b. Artificial neural networks c. Data visualization 2. Computation and Algorithms a. Python 3. Mechanical design a. Rover bogie technology 4. Electronics and Electrical development a. Sensors b. Drivers c. Actuators d. Power supplies Generalized Algorithm for our approach:- 1. Begin all physical system and proceed to boot sequence. 2. Prompt for the user input to set the mode into a. Training b. Prediction 3. If the input is Training a. Get all sensors data with respective to driver inputs in the real time environment or

get commands from remote control through internet. b. Store the data for processing and develop a classifier by feeding data to machine learning algorithm. 4. If the input is prediction a. Get all sensors data from the physical system or get commands from remote control through internet. b. Feed that data to pre trained classifier and predict the direction or angle the physical system should follow. Generalized flow diagram of our approach:-

START

Train Mode

Acquire data from sensor and driver inputs Acquire data from sensors. NO YES

Store the Acquired data and train Machine learning algorithm to generate classifier

STOP User sending commands from internet Predict path to take by feeding data to pre-trained classifier

1.3 Applications :- 1. Autonomous vehicles take quick decision on roads in the situations where response time is very less. This avoids accidents on road and save thousands of people. 2. The travel time or shipment time is much predictable than conventional driving system. The navigation from start to destination points become clearer if we connect whole system into centralized or connected computation. 3. By autonomous vehicles we can navigate autonomously anywhere and get data like temperature, humidity and climate conditions etc., One of the best example for autonomous navigation are agriculture rovers. These rovers navigate through out the field and get crucial data elements like soil fertility, temperature, moisture which helps us understand overall crop health and also individual areas of fields. 4. We can use autonomous navigation and data acquisition techniques in the field of military and army applications to get tracks of enemies and their movements. 5. One of the major applications of autonomous navigation is 3d mapping using LIDAR of the surrounding environment which helps architects to understand pre-historical architectures in detail. One of the examples are reconstruction of roman Colosseum in computer aided design. 6. Recent applications of 3d mapping with autonomous navigation help scientist understand the number of fruits in an orchid without manually counting them. The accuracy of fruit count is more than 93%. 7. There are applications where researcher reconstructs the cities using the data obtained from autonomous vehicles.

8. Now a day’s most of the transportation network companies like UBER, Ola are developing autonomous taxi services which are quicker, accurate and economical to users. .

Fig 1.1 Connected Autonomous vehicles Fig 1.2 UBER Autonomous Taxi CHAPTER 2 LITERATURE SURVEY 2.1 Hardware prototype mechanical design:- Rocker-bogie:

The Rocker-bogie design is used in mars curiosity. Rocker-bogie design has six wheels, 3 on each side. This design has a unique suspension system with no springs or stub axles for each wheel, allowing the rover to climb over obstacles. The maximum height rover can climb and eliminate is twice the wheel’s diameter. Each of the rover’s six wheels has an independent motor. Due to independency in the motors more drag force will be created.

Fig 2.1.1 Rocker-bogie chassis

Fig 2.1.2 Rocker-bogie design stimulations 2.2 Hardware Architecture:-

Fig 2.2.1 In-depth hardware architecture

In-depth views of architecture:-

Fig 2.2.2 Enlarged view 1 of hardware architecture

Fig 2.2.3 Enlarged view 2 of hardware architecture

Physical prototype front view: -

Fig 2.2.4 physical prototype

Description on connections: 1. Arduino connections : Arduino 8, 9 pin to Ultra Sonar Echo, Trig respectively. Arduino 3 pin to Servo motor signal pin (yellow or orange wire). Sonar vcc and servo +5v are connected to power bank port 1 +5v wire (Red). Gnd pin of Sonar and servo motor are connected to Gnd of power bank port 1(Black). 2. Raspberry pi connections: Raspberry pi 37, 35, 33, 31 pins are connected to IN1(input 1), IN2(input 2), IN3(input 3), IN4 (input 4) of motor driver L293d. EN1(enable1) and EN2(enable 2) are connected to pin 2(+5v) for full speed of motor (Voltage range:0v-5v =< speed: 0% to 100%). 12 v battery is connected to +12v power supply. motor1(output1) are connected to left sides motors and motor2 (output2) are connected to right sides motors 3. Common ground both power bank and 12v power supply.

2.3 Hardware Components:- 1. Kit4Curious 6 wheel Drive Curious Chassis: Specifications General Sales Package • learning toy Skillset • Curiosity Building, Creativity & Imagination Number Of Batteries • 0 Batteries Minimum Age • 6 Material • Acrylic Glass Battery Operated • Yes Battery Type • No batteries Rechargeable • No Motors • 6 Motors type • B.O Motors Motors voltage • 9 volts Width • 3 inch Height • 5 inch Depth • 4 inch Weight • .48 g

Table 2.3.1 Specification of vehicle chassis

1. Raspberry pi B+ model: - Specifications

• Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz

• 1GB LPDDR2 SDRAM

• 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE

• Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)

• Extended 40-pin GPIO header

• Full-size HDMI

• 4 USB 2.0 ports

• CSI camera port for connecting a Raspberry Pi camera

• DSI display port for connecting a Raspberry Pi touchscreen display

• 4-pole stereo output and composite video port

• Micro SD port for loading your operating system and storing data

• 5V/2.5A DC power input

• Power-over-Ethernet (PoE) support (requires separate PoE HAT

Raspberry pi is the mostly used prototyping microprocessor and low cost board available in market.

Due its large open source community problems and issues are resolved quickly. Raspberry pi is very compatible with python. Raspberry pi run on full stack Linux operating system.

Other alternative: - Beagle bone, Asus Tinker board, Rock64

Fig 2.3.1 Pin diagram of Raspberry pi

2.

Arduino Uno: - Specifications

• Microcontroller: ATmega328

• Operating Voltage: 5V

• Input Voltage (recommended): 7-12V

• Input Voltage (limits): 6-20V

• Digital I/O Pins: 14 (of which 6 provide PWM output)

•

Analog Input Pins: 6

•

DC Current per I/O Pin: 40 mA

• DC Current for 3.3V Pin: 50 mA

• Flash Memory: 32 KB

of which 0.5 KB used by bootloader

• SRAM: 2 KB (ATmega328)

• EEPROM: 1 KB (ATmega328)

• Clock Speed: 16 MHz

Fig 2.3.2 pin diagram of Arduino Uno

3. Motor driver L293D: - Specifications

• Supply Voltage Range 4.5V to 36V

• 600-mA Output current capability per driver

• Separate Input-logic supply

• It can drive small DC-geared motors, bipolar stepper motor.

• Pulsed Current 1.2-A Per Driver

• Thermal Shutdown

• Internal ESD Protection

• High-Noise-Immunity Inputs

Fig 2.3.3 Pin diagram of L293D

4. Ultra Sonar HC-SR04: - Specifications • Power Supply :+5V DC • Quiescent Current : >2mA • Working Current: 15mA • Effectual Angle: >15° • Ranging Distance : 2cm – 400 cm/1″ – 13ft • Resolution : 0.3 cm • Measuring Angle: 30 degree • Trigger Input Pulse width: 10uS • Dimension: 45mm x 20mm x 15mm

Fig 2.3.4 Ultra sonar pin diagram

5. Servo Motor SG90: - Specifications • Operating Voltage is +5V typically • Torque: 2.5kg/cm • Operating speed is 0.1s/60° • Gear Type: Plastic • Rotation : 0°-180° • Weight of motor : 9gm • Package includes gear horns and screws Wire Number Wire Color Description 1 Brown Ground wire connected to the ground of system 2 Red Powers the motor typically +5V is used 3 Orange PWM signal is given in through this wire to drive the motor

Table 2.3.2 Servo motor connections

Fig 2.3.5 Servo motor pin diagram 2.4 Software Architecture:-

Micro Processor REMOTE SERVER Server running a computational language which receives data from micro processor and process the data to send commands to the vehicles A Micro processor that receives data from a microcontroller regarding sensors and send/receives data from server and process the data to active driver motor, actuator etc.,

Communication via internet using TCP/IP and Socket layer

Serial Communication with a microcontroller Trained classifier file Machine learning Algorithm to train/ predict the path

Micro Controller A Microcontroller that is connected to the sensors. It sends data collected from sensors to micro processor

180° servo motor to rotate ultra sonar to from 0° to 180° Ultra sonic sensor to sense distance vector from 0° to 180°

File system and description: - 1. ULV_pi.py: File Location: - Raspberry pi /home/pi/ This file creates intercommunication between raspberry pi and cloud system. This file acts as server file where we connect our cloud system as client. Step 1: booting up raspberry pi with raspbian operating system. Step 2: Configuring Secure Shell host (SSH) in the raspberry pi and configure a static IP address. Step 3: Copy ULV_pi.py in to the directory /home/pi/ Step 4: Install python external module pyserial using command << pip install pyserial Step 5: Connect to raspberry pi through remote server through SSH. Open terminal in remote server and type the command << ssh pi@ipaddres_of_raspberrypi Enter password Step 7: Always run ULV_pi.py file by connecting Arduino to port 1 (/dev/ttyUSB0). Step 8: To run ULV_pi.py. After connecting to SSH enter the command << python ULV_pi.py 2. ULV_getdata.py: File Location: - Remote server /home/.USER_/Desktop/ULV/ This file is used to get training data from the vehicle by using A, S, D, W keyboard keys of remote server. Step 1: Open terminal and navigate to the folder where ULV_getdata.py file exits (/home/USER_/Desktop/ULV/). Step 2: Open new terminal and Run the ULV_pi.py file by using SSH as mentioned above. Step 3: Run the following command in terminal we used in step 1. << python ULV_getdata.py Step 4: To start Capturing data from vehicle Press G and To stop Capturing press H. Step 5: While in Capturing mode to store data press E and to reset and stop storing data again press E. Pressing E will toggle the state of Storing data. Every time you stopped storing the data stored so far is resets to empty. Step 6: When you are in storing data mode to save the data and exit out press Escape key. This will store the data into rawdata.pkl file. 3. ULV_train.py: File Location: - Remote server /home/.USER_/Desktop/ULV/ This file is used to train and create the classifier on the data from the file rawdata.pkl you obtained from ULV_getdata.py Step 1: Open terminal and navigate to the folder where ULV_train.py and rawdata.pkl file exits (/home/USER_/Desktop/ULV/). Step 3: Run the following command in terminal. << python ULV_train.py Step 4: After running the step 3. It gives the accuracy of training and also generates a classifier trainedclf.pkl file. 4. ULV_move.py: File Location: - Remote server /home/.USER_/Desktop/ULV/ This file is used to run the vehicle autonomously by using the classifier generated from ULV_train.py Step 1: Open terminal and navigate to the folder where ULV_move.py and trainedclf.pkl file exits (/home/USER_/Desktop/ULV/). Step 2: Open new terminal and Run the ULV_pi.py file by using SSH as mentioned above. Step 3: Run the following command in terminal we used in step 1. << python ULV_getdata.py Step 4: To start Autonomous driving press G and To stop press H. File Execution Sequence:

RUN ULV_pi.py Step 1 Step 3

RUN ULV_train.py RUN ULV_move.py RUN ULV_getdata.py Step 2

rawdata.pkl trainedclf.pkl

2.5 Computational Language and library files:- Python: - We have chosen main computational Language as python. Python is light weight and well suits for embedded applications. Python is one of the easiest, yet powerful language and also compatible with most of the processor and controllers available in market. It makes easier to program due to the library file contributions from it open source community. Python have almost all libraries that deal with any application. Python is also used mostly used for data science, computation and research fields. Library files and modules: - We use multiple modules to develop our system in which some of them are in-built and well known python modules. Hence, we concentrate more on external modules which are specified and described one by one below. 1. RPi :- RPi.GPIO is a In-built module of python in Raspbian operating system of microprocessor Raspberry pi. Rpi modules enable us to work with General Purpose Input/Output pins. We use this module to control the movement of the vehicle. Example Usage: import RPi.GPIO as GPIO #importing the module GPIO.setmode(GPIO.board) #set pin mode as board GPIO.setup(37,GPIO.OUT) #setting up pin 37 as output GPIO.output(37,GPIO.HIGH) #write 5v out from pin 2. Pyserial: - Pyserial is a external module which enables the python to access the serial communication with serial device. We use this module to get distance vectors from the arduino microcontroller. Example Usage: Import serial S = serial.Serial("/dev/ttyUSB0",9600) #com (Arduino)=/dev/ttyUSB0 Data = S.read() #Read data from Arduino S.write("some_data") #To write data to Arduino 3. Socket: - In-built python module which enable the developer to implement the communication through socket layer with other devices using protocols like TCP/IP or UDP etc.,. Used to send/receive data from server to microprocessor(Raspberry pi). Example Usage: - Import socket sock = socket.socket() #no-parameter mean default protocol TCP/IP sock.bind((ipaddres, port)) sock.listen(1) client, addr = sock.accept() print addr 4. Pickle: - Pickle module is used for serialization of objects which means storing the python objects into files. We use this module to store distance vectors, trained classifiers into files. Example Usage: - Import pickle D=[1,2,3] F=open('filename','wb') #open file in write mode if not exits create Pickle.dump(D,F) #dumping array D in file F.close() F=open('fiename','rb') #open existed file in read mode Data= pickle.load() #load previously saved array F.close() 5. threading: - In-built python module used for multi-threading. We use multi-threading where we need to send and receive data at same time during training the system. 6. matplotlib : - External module which gives the matlab like plotting features to python. 7. sklearn: - sklearn is the external module which enables developer to perform data science algorithms like machine learning, artificial neural networks etc., 8. pynput:- External library which give us flexibility to trigger/ read an keyboard/mouse events Example Usage:- from pynput.keyboard import Key,Listener def on_press(key): print "pressed key"+chr(key) def on_release(key): print "released key"+chr (key) with Listener(on_press=on_press, on_release=on_release) as listener: listener.join() These are most significant modules we used in the development of the system.

2.6 Machine learning and Artificial neural networks: - Machine learning and Artificial Intelligence are sub branches of data sciences and analytics. Both ML and AI use statistics and probability theory to predict the output. Machine learning algorithms need huge amount of

data to train. Based on the type of learning Machine learning is classified into 3 types. They are 1. Supervised learning 2. Unsupervised learning 3. Reinforcement Learning

1. Supervised learning: - In supervised learning there are set of inputs and set of outputs corresponding to the inputs. In supervised learning the we train the system with both our past input and output data. After that we pass our new input and algorithm predicts the output. Based on the process we classified these algorithms as two types.

a. Classification: In classification process the system work is to classify the data into different classes.

Fig 2.6.1 Classification of circle from crosses

b. Regression: In regression we try to linear equation best represents relation between the data points or direction in which a data is spreading as shown below.

.

Fig 2.6.2 Linear equation best fits the data 2. Unsupervised learning: In unsupervised learning the algorithm only have inputs and there are no corresponding outputs for the input to train the system. But at the same time the input data have pattern which the algorithm have to find and separate the data accordingly.

Fig 2.6.3 Differentiating supervised and unsupervised learning Most Unsupervised algorithm follows the method called data clustering. Clustering: In clustering the data is classified based on distances of points from one another. In Fig 2.3.3 Unsupervised learning all red data point are closer to one another so they are cluster as one class of data by the algorithm. 3. Reinforcement learning:

In this type of learning the system teaches itself by trial and error method. For example if the system is given a task of putting a ball in to basket. The system need to find the best angle to through the ball for which system gives several trial until it reaches the sufficient accuracy.

Fig 2.6.4 Example for Reinforcement learning Algorithms and their classifications:

Fig 2.6.5 Machine Learning algorithms and classification

Sklearn Algorithm Reference Chart :

Fig 2.6.6 Sklearn algorithm chart

2.7 Network and communication protocols: - In this proto type we are using two communication methods. 1. Communication between Raspberry pi and Remote Server. 2. Communication between Raspberry pi and Arduino.

1. Communication between Raspberry pi and Remote Server: - This is based on Internet and IP. Both Raspberry pi and Remote server has IP address to communicate with one another. We use a wireless router from JIO network which was connected to the raspberry pi using Wi-

Fi protocol. Every router has DHCP which allocates IP's to the connected devices in random order. So, to eliminate DHCP we need to configure a static IP address based on MAC (media access control). To access router from outside networks we also need to eliminate Network Address Translation (NAT). To eliminate NAT we need to write rules to forward incoming Packets to our static IP address on desired port.

Fig 2.7.1 Example Network diagram implementing NAT In the above network diagram incoming packets from server 10.1.1.1 are routed to our router with public IP address 200.1.1.1 on port 3212. Router then lookups in the NAT table to forward packets to the internal IP 10.1.1.1 on port 3212.

2. Communication between Raspberry Pi and Arduino: - This Communication is based on Serial protocol which is famous inter device communication. In serial communication we have RX and TX pins of one device connected TX and RX pins of other device respectively. Both devices communicate with the basis of baud rate or bit rate.

Fig 2.7.2 Serial communication Step 1: Connect Arduino with Raspberry pi using serial cable as show in fig 2.2.1. Step 2: Connect to raspberry pi through SSH or open terminal directly if you have HDMI display connection to raspberry pi Step 3: To check the all USB devices on the USB peripherals use the command << lsusb Step 4: If you find Arduino in the List the Arduino has successfully connected to Raspberry pi. Step 5: To know the serial port to which the Arduino is connected use the command << ls /dev/tty* Step 6: This command gives a list of serial ports active in the system and you are likely to found a serial port /dev/ttyAMC0 when you connect Arduino. If you disconnect the Arduino this serial port will not be listed which gives us confirmation that serial port is /dev/ttyAMC0. This port name is feed to pyserial as we discussed in chapter 2.5. CHAPTER 3 EXPERIMENTAL ANALYSIS 3.1 Choosing Machine Learning algorithm and understanding data : We have discussed the process of acquiring data from the vehicle in chapter 2.4 using ULV_getdata.py file. In this chapter we train the vehicle to take right turn whenever it comes up with an obstacle. Let's consider 4 decisions that machine learning algorithm have to take while driving the rover. Decision 1: Forward Decision 2: Reverse Decision 3: Right Decision 4: Left So, the work of the algorithm is to classify the data into forward, reverse and right, left. So, we need to use any one of the classification algorithm in machine learning .Let's have 4 classes' 1,2,3,4 representing forward, reverse and right, left motions of the vehicle. The sensor data we have is input to the algorithm and direction of the vehicle is required output. Since we are manually driving the vehicle while in training mode we can get current direction (forward, reverse, right, left) of the vehicle corresponding to sensor data. So, we have both input and output to the classification algorithm which mean we are dealing with a supervised learning system. So, we need a classification algorithm from supervised learning system. Once lookup machine learning algorithm chart 2.6.6 figure we have in chapter 2.6 which leaves us no choice but to choose these below mentioned algorithms 1. Linear SVC (Support Vector Machine) 2. Naive Bayes (Probability Theory) We will test the accuracy of prediction with each of these algorithms. We will also compare training time and prediction for both algorithms. As we need real time prediction the prediction time should be very less and training time can be more as only train once. We have choose such algorithm with lower prediction time and average training time. Understanding data from

sensor: - The major data we are using is distance vectors. This is the data of Ultrasonar distance sensor at every 30° while rotating from 0° to 180°. [0°, 30°, 60°, 90°, 120°, 150°, 180°, 180°, 150°, 120°, 90°, 60°, 30°, 0° , direction] The above data is array of distances at represented angles. At the end of every sample we will have direction of vehicle at that instant. Since servo motor rotates form left to right and right to left we have angle from 0° to 180° (servo motor clock wise sweep) and 180° to 0° (servo motor anti-clock wise sweep). Feature : [0°, 30°, 60°, 90°, 120°, 150°, 180°, 180°, 150°, 120°, 90°, 60°, 30°, 0°] Class: direction (1 or 2 or 3 or 4) 3.2 Real-time data acquisition: - As we mentioned in chapter 3.1 we will train the vehicle to move right side whenever it found an obstacle. Let's take look on the data collected during this training. Distance Vectors from ultrasonar and corresponding state of the rover class Clock wise sweep of servo Anti Clock wise Sweep of servo

0° 30° 60° 90° 120° 150° 180° 180° 150° 120° 90° 60° 30° 0° 1 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 51 22 19 51 51 51 51 18 19 51 51 1 51 51 51 51 51 51 31 25 25 26 51 51 51 51 1 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 51 21 20 51 51 51 51 51 25 51 51 1 51 51 51 51 51 26 27 27 27 51 51 51 51 1 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 51 24 22 51 51 51 51 21 22 51 51 1 51 51 51 51 51 26 25 25 26 26 30 51 51 51 3 51 51 51 51 51 26 24 24 24 51 51 51 51 1 51 51 51 51 51 51 51 25 25 26 51 51 51 51 1 51 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 51 20 20 21 51 51 24 21 51 51 51 1 51 51 51 51 51 51 27 27 27 27 51 51 51 51 3 51 51 51 51 51 51 17 16 16 17 51 51 51 51 1 51 51 51 51 51 51 51 22 21 51 51 51 51 51 1 51 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 51 19 18 20 51 51 51 20 51 51 51 1 51 51 51 51 51 51 26 25 25 25 51 51 51 51 1 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 51 19 17 18 51 51 21 18 51 51 51 1 51 51 51 51 51 51 25 22 21 22 51 51 51 51 1 51 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 51 24 21 51 51 51 51 20 23 51 51 1 51 51 51 51 51 51 28 27 28 29 51 51 51 51 1 51 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 51 22 21 26 51 51 51 51 51 51 51 1 51 51 51 51 51 51 29 28 28 27 30 51 51 51 3 51 51 51 51 51 51 21 20 19 18 22 51 51 51 1 51 51 51 51 51 51 21 19 18 19 51 51 51 51 1 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 32 28 28 51 51 51 51 24 19 22 51 1 51 51 51 51 51 51 22 20 20 20 51 51 51 1 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 51 32 29 29 51 51 51 51 23 51 51 1 51 51 51 51 51 51 51 29 29 29 51 51 51 51 1 51 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 51 20 20 51 51 51 51 15 14 51 51 1 51 51 51 51 51 51 51 26 25 26 51 51 51 51 1 51 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 51 14 13 17 51 51 51 14 16 51 51 1 51 51 51 51 51 21 19 20 20 20 51 51 51 51 1 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 51 26 25 51 51 51 51 19 17 51 51 1 51 51 51 51 51 51 51 22 21 21 51 51 51 51 1 51 51 51 51 51 51 51 51 51 51 51 51 51 3 51 51 51 51 21 20 51 51 51 51 51 19 51 51 Table 3.2.1 Real-time dataset from prototype In our data class 1 is forward movements and class 3 is right turn of the vehicle. All the distance vectors are features and corresponding classes are group in to two vectors. 1. Feature Vector 2. Class Vector The Above data is sample of dataset of length 320 points. It took 424.63 seconds for collecting the dataset. With increase in data the prediction accuracy also increases. 3.3 Data Analysis: - Dimensionality reduction: - Consider data points A(x,y), B (x,y,z). The difference between two data points is dimensions of the data. Dimensions are axes in the graph. If the Dimensionality increases the number of axes to visualize the data also increases. Human intuition can only grasp up to 3 dimensions i.e. 2d and 3d geometry. If we look at previous chapter the feature vector has dimensionally of 14 which is impossible to visualize by the human brain. Hence there are many algorithms developed for the

dimensionality reduction. These algorithms come under visualization branch in our algorithm chart in fig 2.6.6. There two famous dimensionality reduction algorithms. They are 1. Principle component Analysis 2. t-distributed stochastic neighbor embedding In our analysis we use t-distributed stochastic neighbor embedding for dimensionality reduction. We reduce the data to both 2 dimensions and 3 dimensions. To run dimensionality reduction we have two files which take rawdata.pkl file as input which was generated during data acquisition as mentioned in chapter 2.4. 1. datavisualization2D.py 2. datavisualization3D.py Two run these files follow below steps: - Step 1: Open terminal in remote server and navigate to /home/.../ Desktop/ULV where above two files and rawdata.pkl files exits Step 2: Run the following commands For 2D reduction : << python datavisualization2D.py For 3D reduction : << python datavisualization3D.py Wait for some time while the processing take place. After that you get matplotlib figures where you can easily analyze the data and decide whether the data is in separable nature.

rawdata.pkl

Datavisualization3D.py Datavisualization2D.py

2D graph 3D interactive graph

2 dimensionality reduction: -

Fig 3.3.1 2D reduction of data In the above figure all green points are class 1 (forward movement) points and all the red points are class 2 (Right turn). If we carefully observe the data we can split the area into two half based on data type with a line. By observing the data we can easily say that both classes are separable.

Fig 3.3.2 2D reduction analysis The above figure demonstrates that data can be linearly separable. 3dimensionality Reduction:

Fig 3.3.3 3D reduction of data and analysis The data points can be separable by using a linear plane as shown in figures.

3.4 Training with different algorithms: - After data visualization if the data is linearly separable and contains enough data points start training by feeding distance vectors and corresponding classes to the sklearn SVC or naive bayes as mentioned in chapter 2.4 by running ULV_train.py. Training with SVC :

Training with naive bayes :

Both SVC and naive bayes have almost same training (fitting) time and predicting time. But we can't tell which one is better without discussing about accuracy of the prediction which we will discuss in results. In machine learning we always split data into testing data and training data in supervised classification. Once we completed with training with training dataset. We use testing dataset to find the accuracy of the classifier generated. In the above training condition out of 309 points 15 points are stored for testing purpose. There are many other algorithms

we can try which might give better accuracy and prediction time. Some of them are 1. Random forest 2. K nearest neighbours 3. SGD classifier

CHAPTER 4 RESULT AND DISCUSSIONS 4.1 Accuracy of prediction: The accuracy increases with the increase in data. As we only have moderate dataset. The acceptable accuracy is about 80%. Every time we run the algorithm we randomly shuffle the input feature and classes. So, we get different accuracies each time we run. In this situation for better result we run the algorithm for n time and average the accuracies. We also consider highest accuracy in n times. Training and prediction with SVC: 1. length of training set: 294 length of testing set: 15 Average Accuracy: 93.33333333333333 fitting time : 0.0539999008179 predicting time : 0.0119998455048 highest Accuracy: 100.0

2. length of training set: 263 length of testing set: 46 Average Accuracy: 93.69565217391305 fitting time : 0.0420002937317 predicting time : 0.0340003967285 highest Accuracy: 97.82608695652173

3. length of training set: 155 length of testing set: 154 Average Accuracy: 90.71428571428574 fitting time : 0.0199999809265 predicting time : 0.102999925613 highest Accuracy: 94.15584415584416 As we observe the results with decrease in number of data sample the accuracy also goes down. Training and prediction with naive bayes: 1. length of training set: 294 length of testing set: 15 Average Accuracy: 95.33333333333334 fitting time : 0.00999975204468 predicting time : 0.0150001049042 highest Accuracy: 100.0

2. length of training set: 263 length of testing set: 46 Average Accuracy: 94.7826086956522 fitting time : 0.00999975204468 predicting time : 0.0479996204376 highest Accuracy: 100.0

3. length of training set: 155 length of testing set: 154 Average Accuracy: 93.63636363636365 fitting time : 0.00699973106384 predicting time : 0.150000095367 highest Accuracy: 95.45454545454545

4. length of training set: 62 length of testing set: 247 Average Accuracy: 90.76923076923076 fitting time : 0.00899982452393 predicting time : 0.25600028038 By comparing SVC and Naive bayes. Naive bayes has better accuracy and less prediction time.If you observe last stimulation result you will infer that Naive bayes perform extremely well even with smallest dataset. So, Naive bayes is better algorithm for our application. CHAPTER 5 CONCLUSION By this project we demonstrated the different approach to develop autonomous vehicle using machine learning algorithms. We also compared two major machine learning algorithms SVC and Naive bayes. In our observation Naive bayes has better accuracy, prediction time and training time. We have designed a physical prototype and also acquired precious data. By using remote server we reduced the processing power required for micro processor or on board vehicle which saves money for the automobile companies. We also developed mathematical model for dimensionality reduction of data into 2D and 3D. By using SVC and Naive bayes, we got upto 93.3 percent average accuracy which is pretty good accuracy. In machine learning accuracies more than 80% are considered to be robust algorithms. So, we have generated a robust prediction algorithm for autonomous vehicles.

length of training set: 294 length of testing set: 15 Average Accuracy: 95.33333333333334 fitting time : 0.00999975204468 predicting time : 0.0150001049042 highest Accuracy: 100.0

From above result to predict 15 features it took 0.015 seconds. Then the prediction time for single distance vector in real time is 0.015/15=0.001sec approximately. The sensitivity of our algorithm is 1 millisecond i.e. per every 1 millisecond the vehicle checks the environment to take decisions. The reaction time of the vehicle is also 1 millisecond.

CHAPTER 6 FUTURE SCOPE By using sophisticated distance measuring devices like LIDAR we can implement this system in real-time vehicles. In our prototype we only test classification in to two classes forward and right. But in real-time vehicles we have to consider different steering angles for classification. Each distance vector is classified into certain steering angle. For example if the total angle of steering is about 270 degrees. By consider 1° resolution we will have 270 classes. Autonomous vehicles can be used in situation where driving is repetitive and boring like on heavy traffic highways. Soon the road vehicle network we know are going to extent and world is converting into self driving vehicles.

Fig 6.0.1 LIDAR visualization

We can also use self driving cars for the reconstruction of cities in 3d map and reconstruction of old monuments. 3D mapping has many applications how a days like crop harvesting, image reconstruction etc.

CHAPTER 7 REFERENCES All codes that we discussed in the project are available in the given link https://1drv.ms/f/s!Ahz1wq8tFp282TZwbThA0-Wp4KQ3 Websites to refer: - https://www.raspberrypi.org/ https://www.arduino.cc/ https://scikit-learn.org/stable/documentation.html https://pypi.org/project/pynput/ https://pythonhosted.org/pyserial/ Contact Email: - laskhman.mallidi@gmail.com

Page | 2

Hit and source - focused comparison, Side by Side:

Left side: As student entered the text in the submitted document.
Right side: As the text appears in the source.
