IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 2, February 2021

DOI 10.17148/IJARCCE.2021.10206

Auto-Driving Robot: Powered by Neural Networks

Pratik Gupta[1], Santosh Panchal[2], Mayank Tiwari[3]

Computer Science Engineer, Department of Computer Science & Engineering,
Acropolis Institute of Technology & Research, Mangliya Sadak, Indore-453771, (M.P), India[1]

Mechanical Engineer, Department of Mechanical Engineering2
Medicaps Institute of Technology & Management, AB Rd, Pigdamber, Rau, Indore-453331, (M.P), India[2]

Computer Science Engineer, Department of Computer Science & Engineering,
Swami Keshvanand Institute of Technology, Management & Gramothan, Ram Nagariya Rd, Shivam Nagar, Jagatpura, Jaipur-302017, (R.J), India[3]

Abstract: In this paper, the combination of machine learning & neural networking is described, a self-driving toy robot. The body of the robot is built with Lego Mindstorms. An Android smartphone is used to capture the view in front of the robot. A user first teaches the robot how to drive; this is done by making the robot go around a track a small number of times. The Image data, along with the user action is used to train a Neural Network. At run-time, Images of what is in front of the robot are fed into the neural network and the appropriate driving action is selected. The following vehicle will follow the target (i.e., Front) vehicle automatically. The other application is automated driving during the heavy traffic jam, hence relaxing driver from continuously pushing brake, accelerator or clutch. The Idea described in this paper has been taken from the Google car, defining the one aspect here under consideration is making the destination dynamic. This project showcases the poi of python's libraries, as they enabled me to put together a sophisticated working system in a very short amount of time. Specifically, I made use of the Python Image Library to down sample Images, as Ill as the Pyran neural network library. The robot was controlled using the NXT-python library. This paper further Improves as near technologies Ire used as compared to previous projects, which significantly helps In Improving execution time & runtime memory.

Keywords: auto-driving; self-driving; neural networks; robotics; machine learning; supervised learning; artificial intelligence; python.

I. INTRODUCTION

Artificial Intelligence (AI), *supervised learning*, and *neural networks* represent Incredibly exciting and powerful machine learning-based techniques used to solve many real-world problems. For a primer on machine learning, you may want to read this five-part series that I wrote. While human-like deductive reasoning, Inference, and decision-making by a computer is still a long time away, there have been remarkable gains in the application of AI techniques and associated algorithms. Automated vehicles are technological development in the field of automobiles. Although the automated vehicles are for ease of humankind yet they are the most expensive vehicles. In the paper considering the different features and the cost, on a small scale a three-wheel Vehicular Robotic prototype has been designed that will automatically reach the destination of another vehicle to which It is supposed to follow. I have focused on two applications of an Automated Vehicles here and designed a prototype vehicle for that. The one major issue is during heavy traffic a driver has to continuously push brake, accelerator and clutch to move to destination slowly. I have proposed a solution to relax the driver in that situation by making vehicle smart enough to make decisions automatically and move by maintaining a species distance from vehicles and obstacles around. The second issue is when two vehicles have the same destination but one of the drivers doesn't know Its route. The driver can make his vehicle follow the front vehicle If they are known and share their location to reach the same destination. A three-wheeled Mobile Robot is used for research is given. The Mobile Robot consists of multiple sensors, which helps It to communicate with Google Maps API (Application Program Interface) and makes It determine obstacles in order to follow the route and move smoothly. The Mobile Robot connects

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 2, February 2021

DOI 10.17148/IJARCCE.2021.10206

directly to Google Maps API using GPRS Module, gets route and moves in that direction. While the ultrasonic sensors, which have been used for prototype design, helps to avoid obstacles on run time.

Building a self-driving car requires expensive sensing equipment. For example, the Stanford entry In the DARPA grand challenge had 5 different laser measurement system [4]. It is Interesting to consider, If It is possible to create a self-driving car only using data from a camera. Around 1993, CMU created a learning system called "ALVINN" (Autonomous Land Vehicle in a Neural Network) [3], which could control a testbed vehicle to drive on a variety of road surfaces. ALVINN worked by first "watching" a human driver's response to road conditions. After just 5 minutes of such training data in new situations, ALVINN could be trained to drive on a variety of road surfaces, and at speeds of up to 55 miles per hour. At first blush, it is starting that simply feeding Image data and driver response to train a neural network would lead to a working autonomous vehicle. Earlier this year, David Singleton put up a blog post describing his Irked project – a self-driving RC car [1]. As the project dealt with a small RC vehicle In an Indoor environment, the technique was simpler than that used in ALVINN. We were Inspired by David's post and decided to Independently replicate this project using a Lego Mindstorms robot Instead of an RC car. While we started out with limited experience using Neural Networks, we succeeded in my endeavour to create a self-driving robot that can navigate a track In an Indoor environment. Figure.1 shows the Lego robot in action. The purpose of this paper is to share details of how we utilized a set of Python libraries - NXT-python to control the Lego robot, Python Image Library (PIL) to process camera Images, and the Pyran library to train and use an artificial neural network - to build a self-driving Lego Mindstorms robot.

II. ROBOT CONSTRUCTION

We used the Lego Mindstorms NXT 2.0 set to construct the robot. The set consists of various construction elements, motors and sensors. A key element of the set is a microcomputer called the NXT Intelligent Brick. The NXT brick contains a 32-bIt ARM7 microprocessor, flash memory, a battery holder, USB 2.0 port, supports Bluetooth communication and also has ports for connecting sensors and motors. While the Lego Mindstorms set contains a variety of sensors, we did not utilize any of them for this project. The motors In the Lego set are Interactive Servo Motors. Unlike regular hobbyist motors, these motors can be rotated a specific number of degrees (the Lego motors are precise to 1 degree). The robot we constructed has two Independent tracks (each controlled by a separate motor). The motors are pored and controlled by an NXT brick, which is mounted on top of the tracks. The most challenging part of the robot build was creating a secure holder for my smart phone. While the phone has two cameras (front and back), I made the glass of the phone face backwards. This resulted in an acceptable camera mounting. I took care to have the Images coming from the camera show what is directly in front of the robot and minimize views that are far away. That said, we did not have to spend too much effort in optimizing the camera view. The mount we created also had a mechanism to quickly release the phone, which was useful in debugging, charging the phone, and other activities. Figure.2 shows a close up of the phone mount on the robot, and Figure.3 shows a schematic of the mount.

III. SETUP

All codes (written In Python) run on a Windows 10 PC. The PC communicates with the Lego NXT brick vial Bluetooth. An Android camera phone (Google Nexus S) is attached to the Lego robot. The phone is connected to my home wireless network, as is my PC. Figure.4 shows a diagram of communication between the various components.

IV. DRIVING THE LEGO MINDSTORMS ROBOT

We used the NXT-python library to Interface my PC to the Lego Mindstorms robot. While the NXT brick does possess flash memory to allow programs to reside on the robot Itself, NXT python works by executing on a PC and sending short commands to the NXT brick vial Bluetooth or USB. As I want the robot to be untethered, I make use of Bluetooth. We made use of an experimental class in NXT-python called "Synchronized Motors" that makes the motors controlling the left and right track to move in unison. If care Ire not taken to move the two motors together, the robot could drift to one side when the Intent is to move straight ahead. Ultimately, the key requirement for the robot's motion is consistency. In my Implementation, the robot had three movement options: It could move straight ahead, turn left or turn right. Each action Int on for a short period of time. The right motor (which ran the right-side track) is connected to PORT A of the NXT brick. Analogously, the left motor is connected to PORT B. In NXT-python, we can create a Motor object that

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 2, February 2021

DOI 10.17148/IJARCCE.2021.10206

represents the motor and provides a function like turn () to move the Interactive servo motor to a specific position using a given amount of poi.

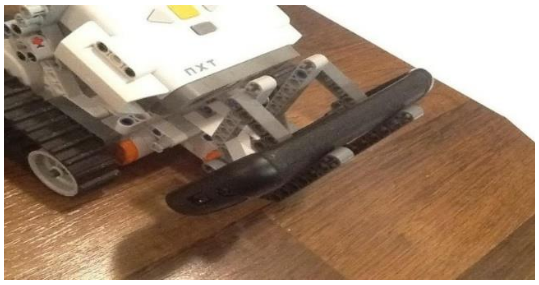

Figure 1: A close up look at the holder mechanism for the Android phone.

Import next
def Airbrick ():
# Define some global to simplify code **global** b, r, l, m, mills, Mrs *# Search and connect to NXT brick (vial BT)* b = nxt.fInd_one_brick() *# Create objects to control motors* r = nxt.Motor(b, nxt.PORT_A) l = nxt.Motor(b, nxt.PORT_B)
# Create objects for synchronIzed motors # I specify the leader, follower and turn ratio m = nxt.SynchronIzedMotors(r,l, 0) mls = nxt.SynchronIzedMotors(l, r, 20) mrs = nxt.SynchronIzedMotors(r, l, 20)
# The first parameter to turn() IndIcates
# 100% or full poIr. To run the motor backwards, # a negative value can be provided. Amt Indicates the # number of degrees to turn. **def** go(dev,amt): dev.turn(100,amt);

To facilitate the collection of training data, I Implemented a “keyboard teleop” mode, wherein I type commands into a python CLI and get my robot to make the appropriate movement (I.e. go straight, go left or go right).
# cmd param is the character typed by the user **def** exec_cmd(cmd):
If cmd == ’f’: go(m,250) **elIf** cmd == ’l’: go(mls,250) **elIf** cmd == ’r’: go(mrs,250) **elIf** cmd == ’x’: b.sock.close()

V. GETTING IMAGES FROM THE CAMERA PHONE

We Initially thought about writing my own app to capture Images from my phone (an Android Nexus S). Hover, we found a free app called IP Inca that allowed me to take snapshots from the phone vial HTTP. Note that the IP address In the URL used to retrieve the Image corresponds to the address assigned to the Android phone by the wireless AP. The list resolution at which I could get Images was 176×144; I processed these Images on the desktop before sending them to the neural network.

Importruleres=urllIb.urlretrieve(’http://192.168.1.12:8080/shot.jpg’)

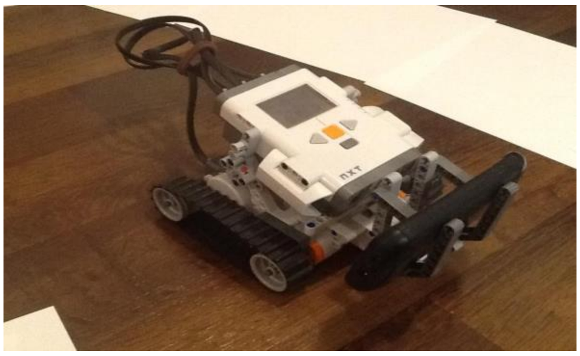

Figure 2: An Image of the Lego robot as it is driving along its course, close up look at the holder mechanism for the Android phone.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 2, February 2021

DOI 10.17148/IJARCCE.2021.10206

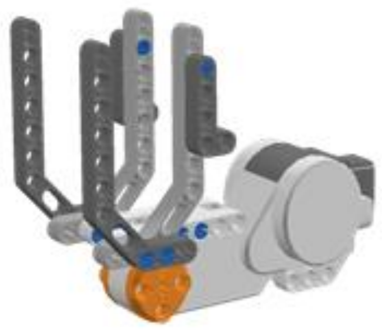

Figure 3: A schematic of the holder mechanism for the Android phone.

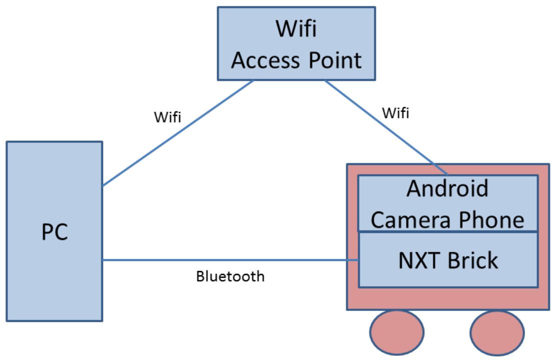

Figure 4: A diagram showing communication between various components.

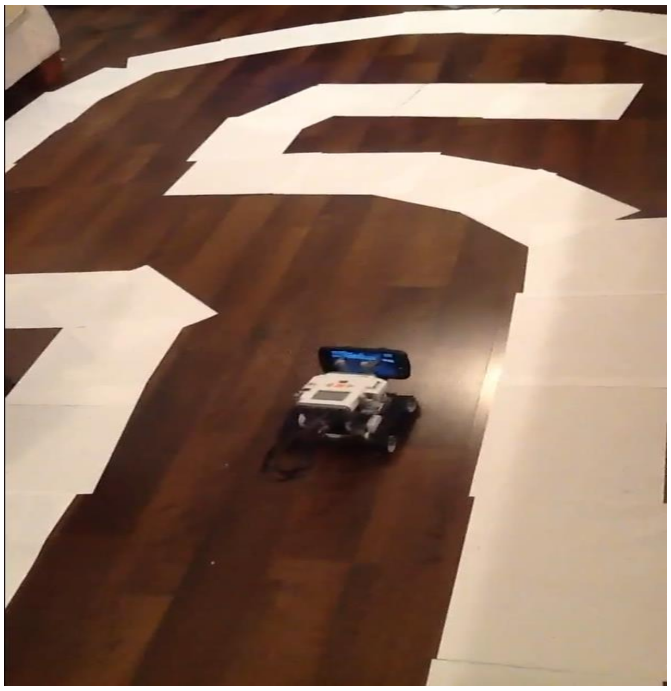

Figure 5: A view of the robot driving on the track.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 10, Issue 2, February 2021
DOI 10.17148/IJARCCE.2021.10206

VI. PROCESSING THE IMAGES ON DESKTOP

We used the Python Imaging Library to first convert the Images from the camera phone to greyscale and then Lower their resolution to 100×100.

```
from PIL import Image
im = Image.open(res[0])
nim = im.convert('L')
nim2 = nim.resize((100,100))
```

We combine the two code fragments above into a function called telepic(), which captures an Image from the Android phone, transforms It and returns the result.

VII. OBTAINING TRAINING DATA

In order to teach the Lego robot how to drive, one must first obtain training data. Each sample of the training data consists of a low-resolution greyscale image showing what is in front of the robot, and a human driver's action in that situation.

```
# This function accepts a command from the
# user via the keyboard, and executes it on the robot
def accept_execute_cmd():
cmd = '';
gotCmd = False;
print "CMD: "
while gotCmd == False:
cmd = getch();
#cmd = raw_input('CMD: ')
if cmd == 'f' or cmd == 'l' or cmd == 'r':
exec_cmd(cmd)
gotCmd = True;
elif cmd == 'x':
b.sock.close()
gotCmd = False;
exit();
print cmd + "\n";
return cmd;
def trainer():
while True:
# download pic from camera and downsample
im=take_pic()
# get cmd from user and run it
cmd = accept_execute_cmd()
# record the image and cmd
record_data(im,cmd)
```

VIII. ENTER THE NUERAL NETWORK

This was the key part of the project. To learn about Neural Networks, I went through Professor Andrew Ng's lectures on Neural Networks, and played around with the assignments on the topic (recognizing hand-written digits using Neural Networks). Luckily, I found the pyBrain project, which provides a very easy interface for using Neural Nets in Python. Similar to David Singleton, I used a three-level network. The first layer had 100×100 nodes. Each input node corresponds to a greyscale image captured from the camera phone. The hidden layer had 64 units (I tried other values, but like David, 64 hidden units worked well for me too). Unlike David, I only had three output units – forward, left and right.

```
from pybrain.tools.shortcuts import buildNetwork
from pybrain.datasets import SupervisedDataSet
from pybrain.supervised.trainers import BackpropTrainer
net = buildNetwork(10000,64,3,bias=True)
ds = SupervisedDataSet(10000,3)
```

IX. TRAINING THE BRAIN

I built a "driving course" in my living room (shown in Figure 5). I drove around the course only 10 times and trained network for about an hour.

```
def train(net,ds,p=500):
trainer = BackpropTrainer(net,ds)
trainer.trainUntilConvergence(maxEpochs=p)
return trainer
```

X. AUTO-DRIVE MODE

The code for auto-drive mode was pretty similar to training mode. I took an image from the camera phone, processed it (greyscale and lowered the res to 100×100) and activated it against the neural net I had trained. The output is one of three commands (forward, left or right), which I send to the same “drive(cmd)” function I used in training mode. I put a short sleep between each command to ensure the robot had enough time to complete its motion.

```
# The following function takes the Neural Network
# and the processed image as input. It returns
# the action selected by activating the neural
# net.
def use_nnet(nnet,im):
cmd = ''
lst = list(im.getdata())
res=nnet.activate(lst)
val = res.argmax()
if val == 0:
cmd = 'f'
elif val == 1:
cmd = 'l'
elif val == 2:
cmd = 'r'
return cmd
# The auto() function takes a trained
# neural network as input, and drives
# the robot. Each time through the loop,
# it obtains an image from the phone (and
# downsamples it). The image data is used
# to activate the Neural Network, the
# output of which is executed on the robot.
def auto(nnet):
while True:
im=take_pic()
cmd=use_nnet(nnet,im)
exec_cmd(cmd)
print "executing .." + cmd
time.sleep(3)
```

XI. THE SELF-DRIVING LEGO MINDSTORMS ROBOT COMES TO LIFE

Mostly. About 2/3 of the time, the robot could go through the entire course without any "accidents". About 1/3 of the time, the robot’s motion takes It to a point where It can only see the track (sheets of white paper). When It gets to that state, it keeps going forward Instead of making a turn. But all-In-all, the results Ire quite alight. Mindstorms robot. The heart of the project is a neural network, which is trained with camera Images of the "road" ahead and user Input. At run time, the same camera Images are used to activate the neural network, and the resulting action is executed on the robot. While a simple vision-based system cannot be expected to perform flawlessly, acceptable performance was achieved.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 2, February 2021

DOI 10.17148/IJARCCE.2021.10206

Our experience suggests that Python programmers can utilize neural networks and camera Images to quickly build other Interesting applications.

XII. PERFORMANCE IMPROVEMENTS

1. The modifications in source code of the updated version of python, used less memory which was not possible in earlier versions of python.
2. The updated python software, uses new libraries hence It becomes a light Alighted code.
3. The size of the Image conversion software, which was used to convert the pictures to greyscale; was further light Eight (<5megabytes) as compared to which the previous software was bulky (>20megabytes).
4. The Bluetooth device used in previous project was v3.0.1 as compared to which the latest Bluetooth technology used was v5.1.0 which Increased the data transfer rate and overall speeds up the process.
5. From the above points, we come to know that overall memory efficiency (i.e., CPU usage, execution time & DRAM usage) Improved & complexity reduced.

XIII. FUTURE SCOPE

There is no denying that Robotic technologies are all set to change the way things are done In the Industries In which they are being Implemented. Entrepreneurs are voicing a similar sentiment and are clearly optimistic about the use of Robotics In various Industrial segments. Robotics is mainly capturing Industries like manufacturing, pharmaceutical, FMCG, packaging and Inspection. A bit of Robotics would also be seen in the healthcare sector primarily in the form of assistive and skill development technologies. The other promising sectors are defence and education.

XIV. CONCLUSION

In this paper, the detail workings of a self-driving Lego Mindstorms robot are described. The heart of the project is a neural network, which is trained with camera Images of the "road" ahead and user Input. At run time, the same camera Images are used to activate the neural network, and the resulting action is executed on the robot. While a simple vision-based system cannot be expected to perform flawlessly, acceptable performance was achieved. Our experience suggests that Python programmers can utilize neural networks and camera Images to quickly build other Interesting applications.

REFERENCES

[1] David Singleton. How I built a neural network controlled self-driving (RC) car! *http://blog.davidsingleton.org/nnrccar*
[2] Iqbal Mohomed. Self-driving Lego Mindstorms Robot *http://slowping.com/2012/self-driving-lego-mindstorms-robot/*
[3] Matthew A. Turk, David G. Morgenthaler, Keith D. Gremban, and Martin Marra. VITS-A Vision System for Autonomous Land Vehicle Navigation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 10(3):342-361, May 1988.
[4] Dean A. Pomerleau. Knowledge-based Training of Artificial Neural Networks for Autonomous Robot Driving, Robot Learning, 1993.
[5] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. Junior: The Stanford entry in the Urban Challenge, Journal of Field Robotics, 25(9):569-597, September 2008.