STM JOURNALS

Journal of Artificial Intelligence Research & Advances
ISSN: 2395-6720
Volume 6, Issue 3
www.stmjournals.com

Forward Collision Avoidance System in a Self-driving Remote Controlled Car: An Application of Machine Intelligence

Zankhana H. Shah**, Tirth Patel, Vatsal Patel, Adhunik Rajput
Department of Information Technology, Birla Vishvakarma Mahavidyalaya Engineering College, Vallabh Vidyanagar, Gujarat, India

Abstract

The paper discusses the experimental vision-based forward collision avoidance system (FORCAS), an autonomous bot car based on machine learning using Raspberry Pi as a processing platform. Using HD camera images as a primary input, Raspberry Pi combines processes and predict path using convolutional neural network and distance estimation models. It distinguishes the stationary objects on a drivable road surface and estimates their distances from a single camera's view. Classification algorithms run on real-time images controls the steering of the car. On the basis of a detected object, it decides to avoid, stop or run over it.

Keywords: *Forward collision avoidance system (FORCAS), Raspberry Pi, Tensorflow, remote controlled car*

****Author for Correspondence*** E-mail: zankhana.shah@bvmengineering.ac.in

INTRODUCTION

The 21st century has come up with advancements in day-to-day life which includes human computer interaction, robotics, and automation using IoT (internet of things), virtual and augmented reality, etc. Advancement in computer technology has made our daily life easy and more accessible. Autonomous cars have gained an interest in recent years as many industries are enthusiastically developing related hardware and software technologies toward fully self-governing driving capability with no human intervention [1]. Driver error is one of the most common cause of traffic accidents due to cell phones, in-car entertainment systems and more traffic. With the number of accidents increasing day-by-day, it has become important to take over the human errors and help the mankind. All of this could come to an end with self-driving cars. An autonomous car (also known as a driverless car or self-driving car) is a robotic vehicle that is designed to travel between destinations without any human intervention.

The motivation of the project is to eliminate the need for hand-coding rules and instead create a system that learns how to drive by observing the coming obstacles on its way. Predicting objects and distance with car is one important part of the entire system.

Forward collision avoidance system (FORCAS) system helps to capture images and store on Raspberry Pi while collecting data as well as controlling and streaming video feed while testing. Making neural network predications locally on a device require enough juice to power-up Raspberry Pi, peripherals and voltage for the motor which has been provided through power-bank battery or DC adapter. In this project, Raspberry Pi, HD camera and many open source software are used to control car, measure distance and recognize objects. It uses supervised learning techniques to decide the path and also to distinguish the stationary objects on a drivable road surface [2]. Figure 1 shows the remote controlled car created in this project.

Fig. 1: *FORCAS System.*

IMPLEMENTATION

Remote Controlled Car

The remote controlled car has been made up by various hardware as listed in Table 1 and Figure 2.

Table 1: *Hardware Used to Make a Remote Controlled Car.*

Logitech C270 USB camera	Breadboard
Raspberry Pi 3-B + case	Resistors, diodes
Toy car	External rechargeable Battery
Dual bridge motor driver IC	32GB MicroSD card
Ultrasonic Module	Soldering equipment, DMM
Copious amounts of cables	

FORCAS Methodology

Forward collision avoidance system (FORCAS) methodology includes two parts of the system, namely remote vehicle and parallel computing machine. They both are connected through Raspberry Pi and GPU to transmit and receive data as per the requirement. Figure 3 describes the methodology pictorially.

System Environment Configuration and Amazon Web Services Setup

NOOBS, a Unix-based file system is chosen as the operating system for Raspberry Pi which was loaded on an 8 GB+ memory card through image writing tool Etcher. As Raspberry Pi has been used as headless, initially need to connect it to a display and keyboard in order to configure various file systems and add scripts. Amazon Web Services EC2 p2.xlarge instance for GPU computation which makes training extremely efficient. Training on GPU takes around 12 hours. To make recovery from unexpected failures easier, we used Tensorflow's checkpoint feature to be able to start and stop models [3].

Hardware Setup

The circuitry inside the bot car has been implemented in the way mentioned in Figure 2, which includes powering the L293D motor driver IC through an external power source [4]. The ultrasonic sensor with a voltage divider circuit using resistors and diode has been implemented in order to get inputs and output signals from ultrasonic in range of 0 to 3.33 V which does not damage the GPIO input-output pins [5]. We decided to implement circuit on a breadboard so that it can be modified while trying and testing. Power to Raspberry Pi Model 3 has been supplied by a 5 V, 2 A power-bank.

This bot car works on bang-bang control as it is using simple DC motors. Bang-bang control is a type of control framework that mechanically or electronically turns something on or off when a coveted target has been attained [6]. This project adapted a geometry model of detecting distance to an object using monocular vision method proposed by Ji et al. [7].

TCP Server, Image and Video Transmission

Raspberry Pi while collecting data handles multiple tasks of getting inputs from the computer for steering, capturing and storing image, getting video feedback to the computer. For controlling steering of the bot, we used L293D IC which was connected to Pi and was sending signals to turn left, right or forward. Control in Pi was implemented by a multithreaded script which used Pynput library [8]. This library allows us to control and monitor input devices and hence being controlled remotely via "w", "a", "d" keys of a keyboard. It was similar to basic server-client programming. While using USB camera, Open CV library we used to capture and store images. For video feed, we used FFmpeg library. We installed FFmpeg into Raspberry Pi then broadcasted video to local IP which can be accessed through Computer/Phone/ Tablet [9].

Data Collection Methods

Object detection data were obtained by advanced google search and running a script and extension to download all the photos available until the script is stopped. During the while data collection process one needs to take care of the type of data as well as various methods of collecting in order to make the model extremely efficient and great data only leads to great ML model. Overfitting and Under-fitting occur mainly due to the inadequate care during data collection, annotation and preprocessing.

The total data collected had to be in a huge amount, in size of GBs. We also took care of

training data as well as testing. Dataset is divided such as 80% are used for training and 20% for testing. Hence, we opened for holdout method which is the simplest kind of cross-validation. In this method, the evaluation may depend heavily on which data points end up in the training set

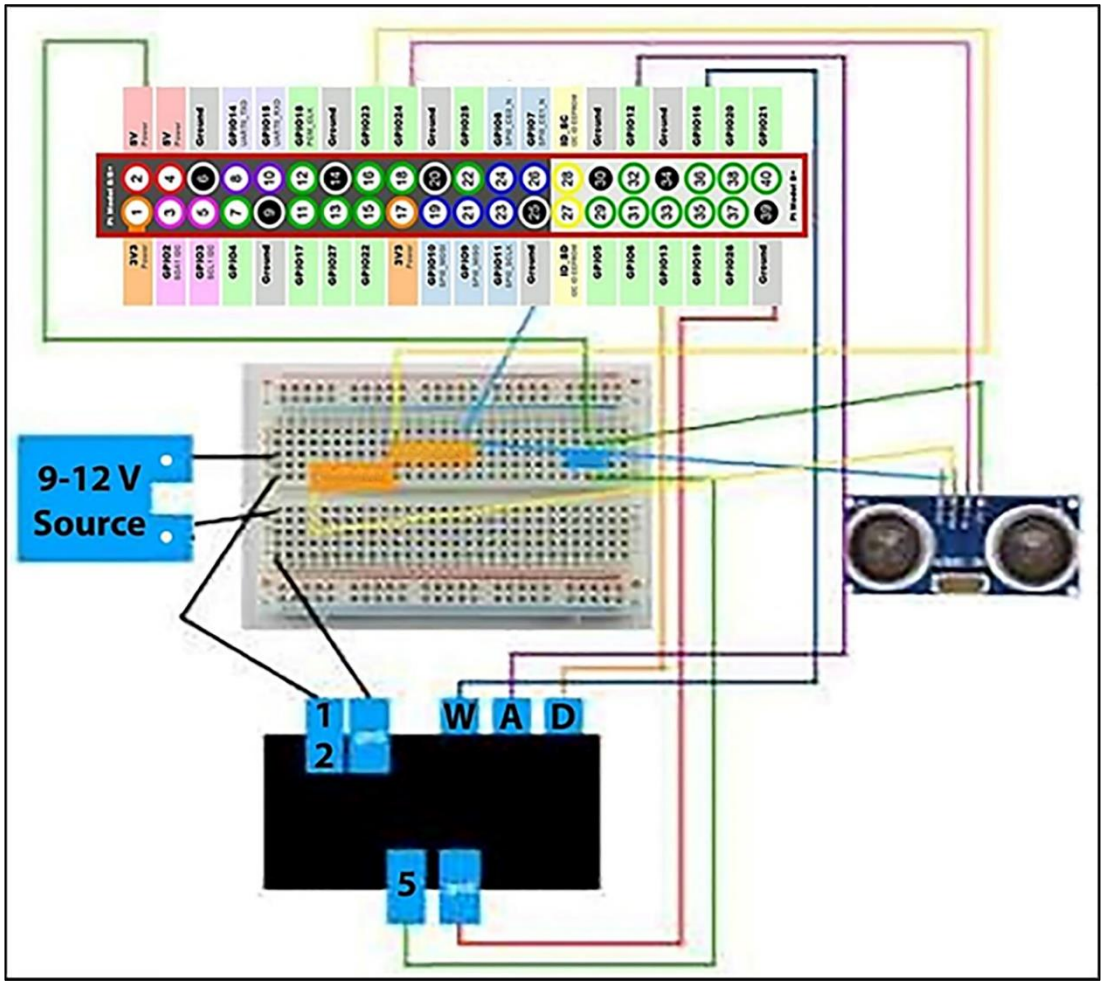

Fig. 2: *Circuitry Inside Remote Controlled Car.*

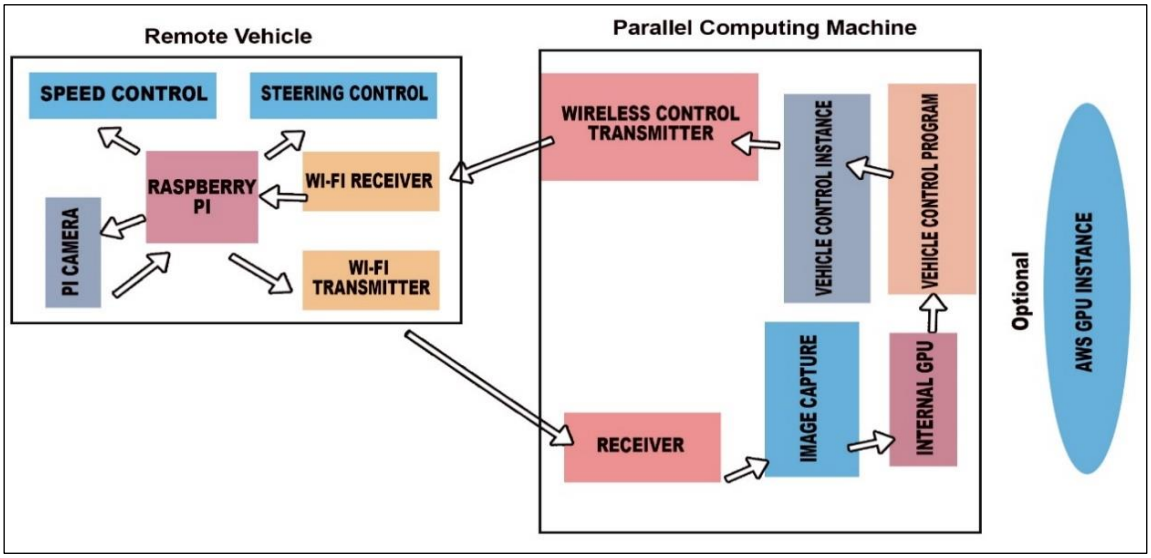

Fig. 3: *FORCAS Methodology.*

and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made [10].

During data collection, photos of a single day were stored in a folder which carries a .csv file also mentioning the key pressed, time stamp and distance recorded on the ultrasonic sensor as shown in Figure 4.

Data collection was made by laying paper on the smooth ground and running bot car over simple track as well as on track with obstacles as shown in Figure 4. We used paper as it was easy to modify track for multiple iterations and can be easily transferred to another place. To increase the amount of data collected and to also cover many possible situations like different brightness, shadow or object is in different situations, we have done the data augmentation process, using Keras and OpenCV library.

Camera and ultrasonic module were pretty portable instead rigidly stick to the body. Data Collection with inclined cameras and ultrasonic will also be helpful in order to make turn efficiently [11]. These datasets train car to come back on track if it moves out of paper (Figure 5). Moreover, it does not make model over fit to a singular environment [12, 13].

Data Cleaning and Data Preprocessing

Data cleaning is required to remove ambiguous as well as unwanted data and last chance to remove mistakes while data collection as the data set gets bigger day-by-day. It is really difficult to make changes in the later stage as the model would have started training on such inconsistent images.

In the process of data preprocessing, we opted for difference of two images, and hence, the Numpy array in model will lighter and train faster. To increase the amount of data collected and to also cover many possible situations like different brightness, shadow or object is in different situations, we have done the data augmentation process, using Keras and OpenCV library, as shown in Figure 6.

controls.csv Open with Atom

timestamp	Images	controls	distance
2018-03-05 21:57:16.502997	2018-03-05 21:57:15.897384	w	76.38
2018-03-05 21:57:20.207657	2018-03-05 21:57:19.614092	s	71.72
2018-03-05 21:57:23.711935	2018-03-05 21:57:22.866729	q	79.69
2018-03-05 21:57:27.906588	2018-03-05 21:57:27.056770	e	90.9
2018-03-05 21:57:32.232756	2018-03-05 21:57:31.368773	q	71.13
2018-03-05 21:57:36.482755	2018-03-05 21:57:35.619832	e	79.05
2018-03-05 21:57:41.594333	2018-03-05 21:57:40.728153	q	65.48

Fig. 4: *Control CSV.*

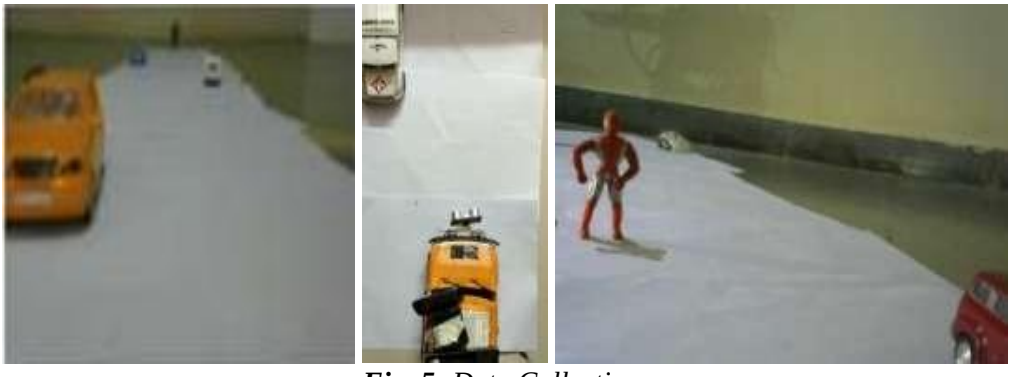

Fig. 5: *Data Collection.*

Fig. 6: *Original and Augmented Images.*

DESIGNING ARCHITECTURE OF NEURAL NETWORK AND MODEL TRAINING ON AMAZON WEB SERVICES

Our neural network consists of 10 layers, which includes five convolution layers, four fully connected layers, and the output layer. We trained our network to reduce the categorical cross-entropy loss between the output of either data collected by a human or artificially created synthesized data and the output of the network. Input size of the image is 66*200.

First five layers are convolution layers, whose purpose is to extract the features of the images and were decided after series of experimenting with a model. We have used 5*5 kernel size and 2*2 stride for the first three layers and 3*3 kernel size with no stride for next two convolution layers. We have used an Adam optimizer with 10 learning rate for minimizing the loss. Next four layers are fully connected layers that are used to get the output class from the three classes (forward, left, right). The purpose of these layers is to act as a controller to steer the car. We have also used dropouts to avoid over fitting the training data. First nine layers contain ReLu optimizing function.

Output layer consists of three neurons trained using Softmax activation function, as we have a data with mutually exclusive classes and categorical cross-entropy loss.

Figure 7 shows the activations of the first feature map layer for an example input. The feature map activations clearly show the outline of the road and it is also able to detect the object. This demonstrates that the CNN (convolutional neural network) learned to detect useful road features on its own, i.e., with only the human steering inputs. We never explicitly trained it to detect the outlines of roads or the objects.

The system is tested for self-created data set consisting of obstacles like small toy cars and statues of super heroes considering white paper as a road.

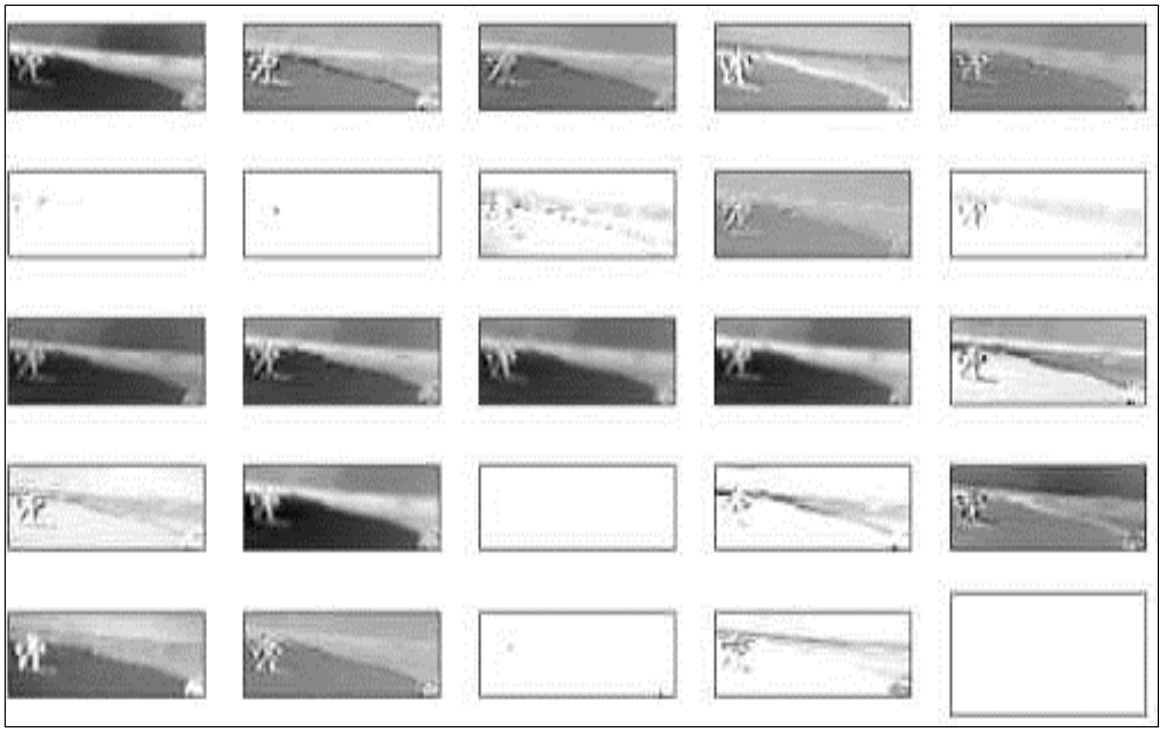

Fig. 7: *Object Detected using Neural Network.*

CONCLUSION

We presented FORCAS system which can be implemented on cars or drones that serve the purpose as a research platform and help learn state-of-the-art artificial intelligence technology. Alternatively, the layers of neural net serve a similar purpose as in real life autonomous car. This model can also be used as a low-cost education and research platform for Machine Learning Algorithms and Control Engineering. Despite the complexity, FORCAS uses low-cost Raspberry Pi Computation. We additionally assessed other, more powerful, embedded computing platforms to better understand achievable real-time performance for FORCAS machine and the effect of figuring equipment structures. However, shared resource contention remains an important issue that must be considered in applying machine-learning models to shared memory based embedded computing platforms. As future work, we plan to apply techniques that help achieve a real-time performance of the system. We also plan to improve the prediction accuracy by feeding more data and upgrading the remote controlled car hardware platform to enable more precise steering angle control.

REFERENCES

1. Pannu GS, Ansari MD, Gupta P. Design and Implementation of Autonomous Car using Raspberry Pi, *IJCA*. 2015, 113(9): 22–29p.
2. Paul A, Chauhan R, Srivastava R, Baruah M. Advanced Driver Assistance Systems, *SAE Technical Paper 2016-28-0223*, 2016. Doi: https://doi.org/10.4271/2016-28-0223.
3. Tensorflow (2019). tensorflow/examples. [online] Available from https://www. tensorflow.org/deploy/distributed [accessed on November 2019]
4. Instructables (2017). DC Motor Control With Raspberry Pi and L293D. [online] Instructables. https://www.instructables.com /id/DC-Motor-Control-With-Raspberry-Pi-and-L293D/ [accessed on November 2019].
5. The Pi Hut. (2019). HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi. [online] Available from https://thepihut.com/blogs/raspberry-pi-tutorials/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi [accessed on November 2019].
6. Sonneborn L, Van Vleck F. The Bang-Bang Principle for Linear Control Systems. *SIAM J Control.* 1965; 2: 151–159p.
7. Chu Ji, Guo Li, Wang, Study on Method of Detecting Preceding Vehicle Based on Monocular Camera, *IEEE Intelligent Vehicle Symposium*, IEEE, Parma, Italy, 2004.
8. PyPI. (2019). pynput. [online] Available from https://pypi.org/project/pynput/ [accessed on November 2019].
9. Hackster.io (2107). Rasbperry Pi-FFmpeg Install and Stream to Web. [online] Available from https://www.hackster.io/whitebank/rasbperry-pi-ffmpeg-install-and-stream-to-web-389c34 [accessed on November 2019].
10. Jeff Schneider. (1997). Cross Validation. [online] Available from https://www.cs.cmu. edu/~schneide/tut5/node42.html [accessed on November 2019].
11. Giusti A, Guzzi J, Ciresan DC, He F, Rodrï Guez JP, Fontana F, Faessler M, Forster C, Schmidhuber J, Di Caro G, Scaramuzza D, Gambardella LM. A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots, *IEEE Robot Autom Lett. (RA-L)*, 2016, 1(2): 661–667p.
12. Coursera.org (2019). Machine Learning. Available from https://www.coursera.org/learn/machine-learning [accessed on November 2019].
13. Memon KR, Memon S, Memon B, Memon AR, Syed MZAS. Real time Implementation of Path planning Algorithm with Obstacle Avoidance for Autonomous Vehicle, *International Conference on Computing for Sustainable Global Development*, 2016, New Delhi, India.

Cite this Article

Zankhana H. Shah, Tirth Patel, Vatsal Patel, Adhunik Rajput. Forward Collision Avoidance System in a Self-driving Remote Controlled Car: An Application of Machine Intelligence. *Journal of Artificial Intelligence Research & Advances.* 2019; 6(3): 13–18p.