RESEARCH ARTICLE   OPEN ACCESS

Based on machine learningAutonomous car using raspberry-pi.

K.N.V.Satyanarayana, B.Tapasvi, P.KanakaRaju, G.RameshBabu
Assistant professors,E.C.E.Department,S.R.K.R.engineeringcollege,Autonomous China Amiram,

ABSTRACT
In this paper we discuss about the design of Autonomous car based on machine learning using raspberry pi. Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Using this concept of machine learning, a car can be automated (self driving). We train the car with specific images and whenever it detects the trained images, it operates according to the trained instruction. The microcontroller used in the car is raspberry pi which is used to control the L298 driver, ultrasonic sensor and the raspberry pi camera. We use different components like pi camera which is used to train and detect the objects, L298 driver which operates the dc motor and the ultrasonic sensor to calculate the distance. This autonomous car is the prototype to the self driving cars which is the present growing advanced technology in the present scenario.
Key Terms: Raspberry Pi, Picamera, Computer, Numpy, OpenCV, Pygame, PiSerial

----

I.   INTRODUCTION

A scaled down version of self-driving system using a RC car, Raspberry Pi and open source software. The system uses a Raspberry Pi with a camera and an ultrasonic sensor as inputs, a processing computer that handles steering, object recognition (stop sign and traffic light) and distance measurement.

I.Machine learning:

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can change when exposed to new data. The process of machine learning is similar to that of data mining. Both systems search through data to look for patterns. However, instead of extracting data for human comprehension -- as is the case in data mining applications -- machine learning uses that data to detect patterns in data and adjust program actions accordingly. Machine learning algorithms are often categorized as being supervised or unsupervised. Supervised algorithms can apply what has been learned in the past to new data. Unsupervised algorithms can draw inferences from datasets.Facebook's News Feed uses machine learning to personalize each member's feed. If a member frequently stops scrolling in order to read or "like" a particular friend's posts, the News Feed will start to show more of that friend's activity earlier in the feed. Behind the scenes, the software is simply using statistical   analysis and predictive   analytics to identify patterns in the user's data and use to patterns to populate the News Feed. Should the member no longer stop to read, like or comment on the friend's posts, that new data will be included in the data set and the News Feed will adjust accordingly.

II.   NEURAL NETWORKS

ML implements feed-forward artificial neural networks or, more particularly, multi-layer perceptrons (MLP), the most commonly used type of neural networks. MLP consists of the input layer, output layer, and one or more hidden layers. Each layer of MLP includes one or more neurons directionally linked with the neurons from the previous and the next layer. The example below represents a 3-layer perceptron with three inputs, two outputs, and the hidden layer including five neurons:
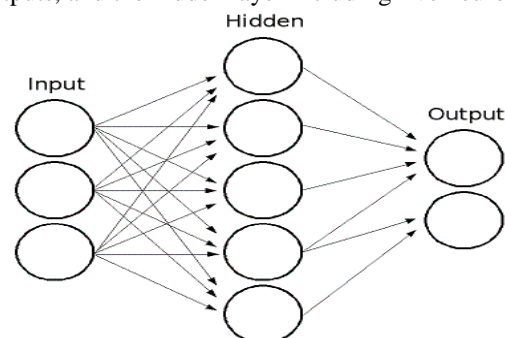
All the neurons in MLP are similar. Each of them has several input links (it takes the output values from several neurons in the previous layer as

----

input) and several output links (it passes the response to several neurons in the next layer). The values retrieved from the previous layer are summed up with certain weights, individual for each neuron, plus the bias term. The sum is transformed using the activation function f that may be also different for different neurons.
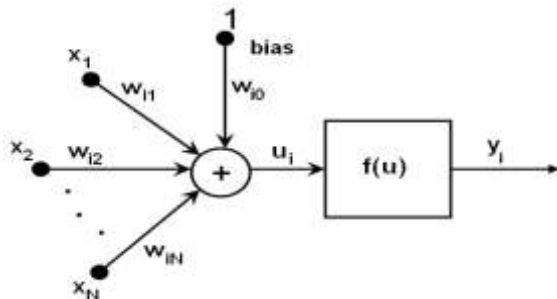
In other words, given the outputs x_j of the layer n, the outputs y_i of the layer $n+1$ are computed as:

$$u_i = \sum_j (w_{i,j}^{n+1} * x_j) + w_{i,bias}^{n+1}$$

$$y_i = f(u_i)$$

Different activation functions may be used. ML implements three standard functions:
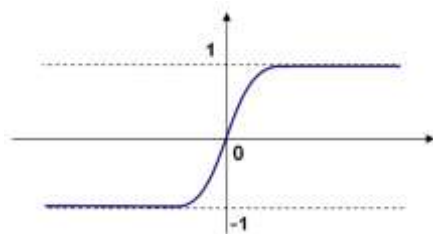
- Identity function (CvANN_MLP::IDENTITY): $f(x) = x$

- Symmetrical sigmoid (CvANN_MLP::SIGMOID_SYM): $f(x) = \beta * (1 - e^{-\alpha x})/(1 + e^{-\alpha x})$, which is the default choice for MLP. The standard sigmoid with $\beta = 1, \alpha = 1$ is shown below:

- Gaussian function (CvANN_MLP::GAUSSIAN): $f(x) = \beta e^{-\alpha x * x}$, which is not completely supported at the moment.

In ML, all the neurons have the same activation functions, with the same free parameters (α, β) that are specified by user and are not altered by the training algorithms.

So, the whole trained network works as follows:

1. Take the feature vector as input. The vector size is equal to the size of the input layer.
2. Pass values as input to the first hidden layer.
3. Compute outputs of the hidden layer using the weights and the activation functions.
4. Pass outputs further downstream until you compute the output layer.

5. **Self Driving RC Car**
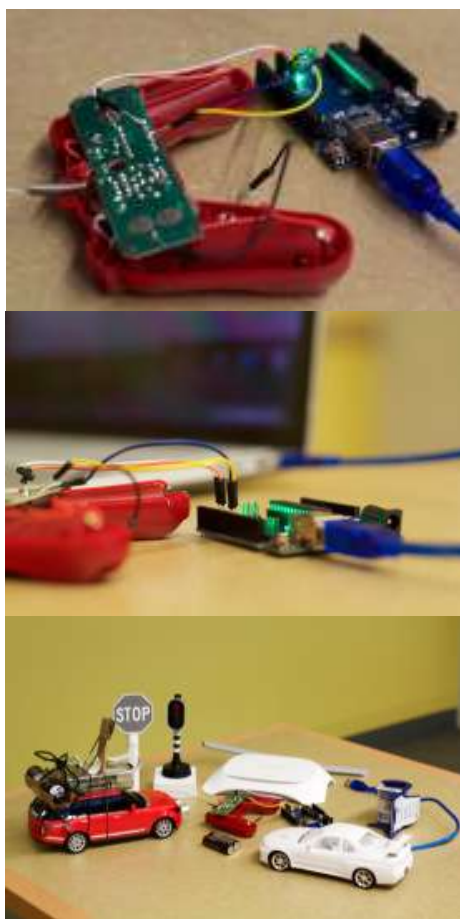6. Python + OpenCV Neural Network + Haar-Cascade Classifiers
7. **Objective**
8. Modify a RC car to handle three tasks: self-driving on the track, stop sign and traffic light detection, and front collision avoidance.
9. **System Design**
10. The system consists of three subsystems: input unit (camera, ultrasonic sensor), processing unit (computer) and RC car control unit.

Input Unit
A Raspberry Pi board (model B+), attached with a pi camera module and an HC-SR04 ultrasonic sensor is used to collect input data. Two client programs run on Raspberry Pi for streaming color video and ultrasonic sensor data to the computer via local Wi-Fi connection. In order to achieve low latency video streaming, video is scaled down to QVGA (320×240) resolution.

Processing Unit
The processing unit (computer) handles multiple tasks: receiving data from Raspberry Pi, neural network training and prediction(steering), object detection(stop sign and traffic light), distance measurement(monocular vision), and sending instructions to Arduino through USB connection.

TCP Server
A multithread TCP server program runs on the computer to receive streamed image frames and ultrasonic data from the Raspberry Pi. Image frames are converted to gray scale and are decoded into numpy arrays.

Neural Network
One advantage of using neural network is that once the network is trained, it only needs to load trained parameters afterwards, thus prediction can be very fast. Only lower half of the input image is used for training and prediction purposes. There are 38,400 (320×120) nodes in the input layer and 32 nodes in the hidden layer. The number of nodes in the hidden layer is chosen fairly arbitrary. There are four nodes in the output layer where each node corresponds to the steering control instructions: left, right, forward and reverse respectively (though reverse is not used anywhere in this project, it's still included in the output layer).
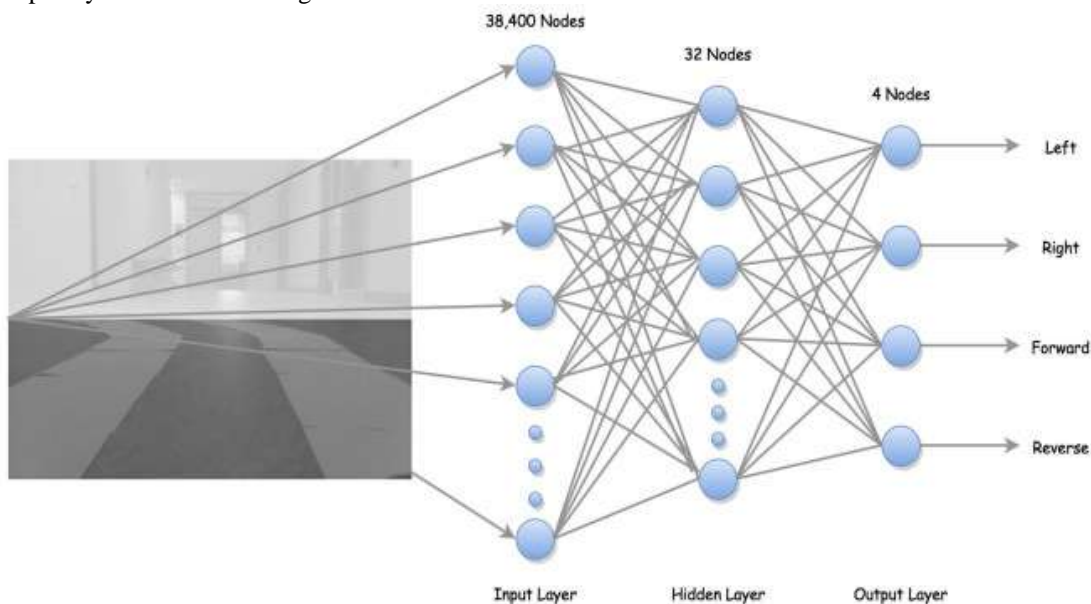
Below shows the training data collection process. First each frame is cropped and converted to a numpy array. Then the train image is paired with train label (human input). Finally, all paired image data and labels are saved into anpz file. The neural network is trained in OpenCV using back propagation method. Once training is done, weights are saved into a xml file. To generate predictions, the same neural network is constructed and loaded with the trained xml file.
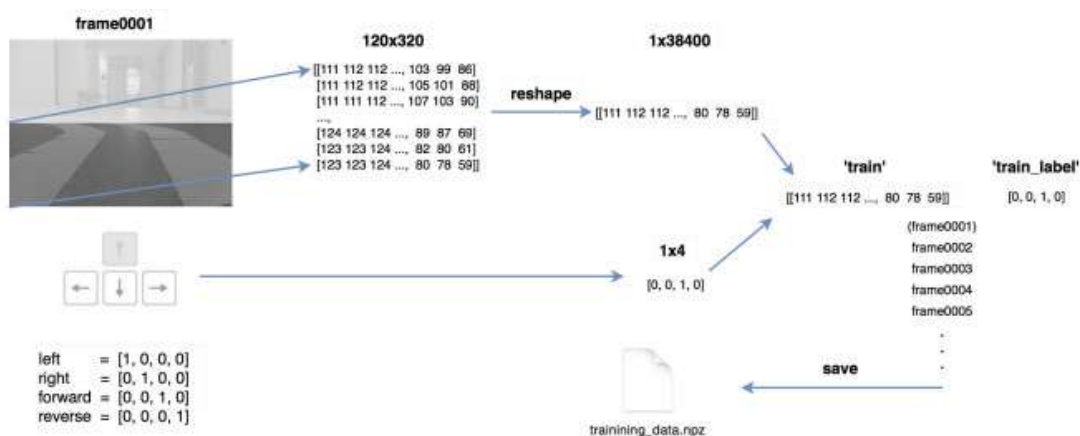
Object Detection

This project adapted the shape-based approach and used Haar feature-based cascade classifiers for object detection. Since each object requires its own classifier and follows the same process in training and detection, this project only focused on stop sign and traffic light detection.

OpenCV provides a trainer as well as detector. Positive samples (contain target object) were acquired using a cell phone, and were cropped that only desired object is visible. Negative samples (without target object), on the other hand, were collected randomly. In particular, traffic light positive samples contains equal number of red traffic lights and green traffic light. The same negative sample dataset was used for both stop sign and traffic light training. Below shows some positive and negative samples used in this project.

Type	Number of positive sample	Number of negative sample	Sample size (pixels)
Stop sign	20	400	25x25
Traffic light	26	400	25x45

To recognize different states of the traffic light(red, green), some image processing is needed beyond detection. Flowchart below summarizes the traffic light recognition process.

Firstly, trained cascade classifier is used to detect traffic light. The bounding box is considered as a region of interest (ROI). Secondly, Gaussian blur is applied inside the ROI to reduce noises. Thirdly, find the brightest point in the ROI. Finally, red or green states are determined simply based on the position of the brightest spot in the ROI.

Distance Measurement
Raspberry Pi can only support one pi camera module. Using two USB web cameras will bring extra weight to the RC car and also seems unpractical. Therefore, monocular vision method is chosen.
This project adapted a geometry model of detecting distance to an object using monocular vision method proposed by Chu, Ji, Guo, Li and Wang (2004).
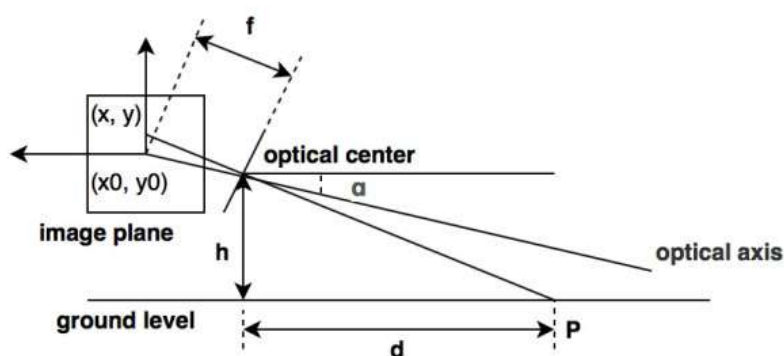
P is a point on the target object; d is the distance from optical center to the point P. Based on the geometry relationship above, formula (1) shows how to calculate the distance d. In the formula (1), f is the focal length of the camera; ∂ is camera tilt angle; h is optical center height; (x0, y0) refers to the intersection point of image plane and optical axis;

(x, y) refers to projection of point P on the image plane. Suppose O1 (u0,v0) is the camera coordinate of intersection point of optical axis and image plane, also suppose the physical dimension of a pixel corresponding to x-axis and y-axis on the image plane are dx and dy. Then:

$$d = h/\tan\left(\partial + \arctan\left((y - y_0)/f\right)\right) \qquad (1)$$
$$u = \frac{x}{dx} + u_0 \quad v = \frac{y}{dy} + v_0 \qquad (2)$$

Let $x_0 = y_0 = 0$, from (1) and (2):

$$d = h/\tan\left(\alpha + \arctan\left((v - v_0)/a_y\right)\right), \qquad (a_y = f/dy) \qquad (3)$$

v is the camera coordinates on y-axis and can be returned from the object detection process. All other parameters are camera's intrinsic parameters that can be retrieved from camera matrix.
OpenCV provides functions for camera calibration. Camera matrix for the 5MP pi camera is returned after calibration. Ideally, a_x and a_y have the same value. Variance of these two values will result in non-square pixels in the image. The matrix below indicates that the fixed focal length lens on pi camera provides a reasonably good result in

handling distortion aspect. Here is an interesting article discussing the focal length of pi camera with stock lens and its equivalent to 35mm camera .

$$\begin{bmatrix} a_x = 331.7 & 0 & u_0 = 161.9 \\ 0 & a_y = 332.3 & v_0 = 119.8 \\ 0 & 0 & 1 \end{bmatrix}$$

The matrix returns values in pixels and h is measured in centimeters. By applying formula (3), the physical distance d is calculated in centimeters.

RC Car Control Unit

The RC car used in this project has an on/off switch type controller. When a button is pressed, the resistance between the relevant chip pin and ground is zero. Thus, an Arduino board is used to simulate button-press actions. Four Arduino pins are chosen to connect four chip pins on the controller, corresponding to forward, reverse, left and right actions respectively. Arduino pins sending LOW signal indicates grounding the chip pins of the controller; on the other hand sending HIGH signal indicates the resistance between chip pins and ground remain unchanged. The Arduino is connected to the computer via USB. The computer outputs commands to Arduino using serial interface, and then the Arduino reads the commands and writes out LOW or HIGH signals, simulating button-press actions to drive the RC car.

III. RESULTS

Prediction on the testing samples returns an accuracy of 85% compared to the accuracy of 96% that the training samples returns. In actual driving situation, predictions are generated about 10 times a second (streaming rate roughly 10 frames/s).

Haar features by nature are rotation sensitive. In this project, however, rotation is not a concern as both the stop sign and the traffic light are fixed objects, which is also a general case in real world environment.

For distance measurement aspect, the ultrasonic sensor is only used to determine the distance to an obstacle in front of the RC car and provides accurate results when taking proper sensing angle and surface condition into considerations. On the other hand, Pi camera provides "good enough" measurement results. In fact, as long as we know the corresponding number to the actual distance, we know when to stop the RC car. Experimental results of detecting distance using pi camera are shown as below:

Order	1	2	3	4	5	6	7
Actual value (cm)	15	20	25	30	35	40	45
Measured value (cm)	15.5	19.7	24.1	27.5	32.0	35.2	39.0
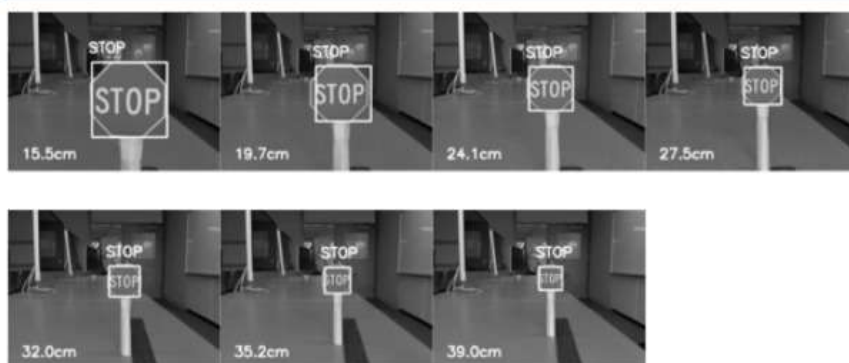
In this project, the accuracy of distance measurement using monocular vision approach could be influenced by the following factors: (1) errors in actual values measurement, (2) object bounding box variations in detecting process, (3) errors in camera calibration process, (4) nonlinear relationship between distance and camera coordinate: the further the distance, the more rapid change of camera coordinate, thus the greater the error.Overall, the RC car could successfully navigate on the track with the ability to avoid front collision, and respond to stop sign and traffic light accordingly.

REFERENCES

[1]. OpenCV Documentation - Haar Feature-based Cascade Classifier for Object Detection
[2]. OpenCV Documentation - Cascade Classifier Training
[3]. NaotoshiSeo - Tutorial: OpenCVhaartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features)
[4]. Material for NaotoshiSeo's tutorial
[5]. OpenCV Answers - "about traincascadeparemeters, samples, and other…"
[6]. OpenCV Answers - "memory consumption while training > 50GB"
[7]. David J Barnes on Robotics & Mechatronics - OpenCVHaarTraining - Object Detection with a Cascade of Boosted Classifiers Based on Haar-like Features - Part I
[8]. David J Barnes on Robotics & Mechatronics - OpenCVHaarTraining - Object Detection with a Cascade of Boosted Classifiers Based on Haar-like Features - Part II
[9]. github.com/foo123/HAAR.js
[10]. github.com/mtschirs/js-objectdetect
[11]. github.com/inspirit/jsfeat
[12]. Computer Vision Software - FAQ: OpenCVHaartraining
[13]. StackOverflow - haar training OpenCV assertion failed