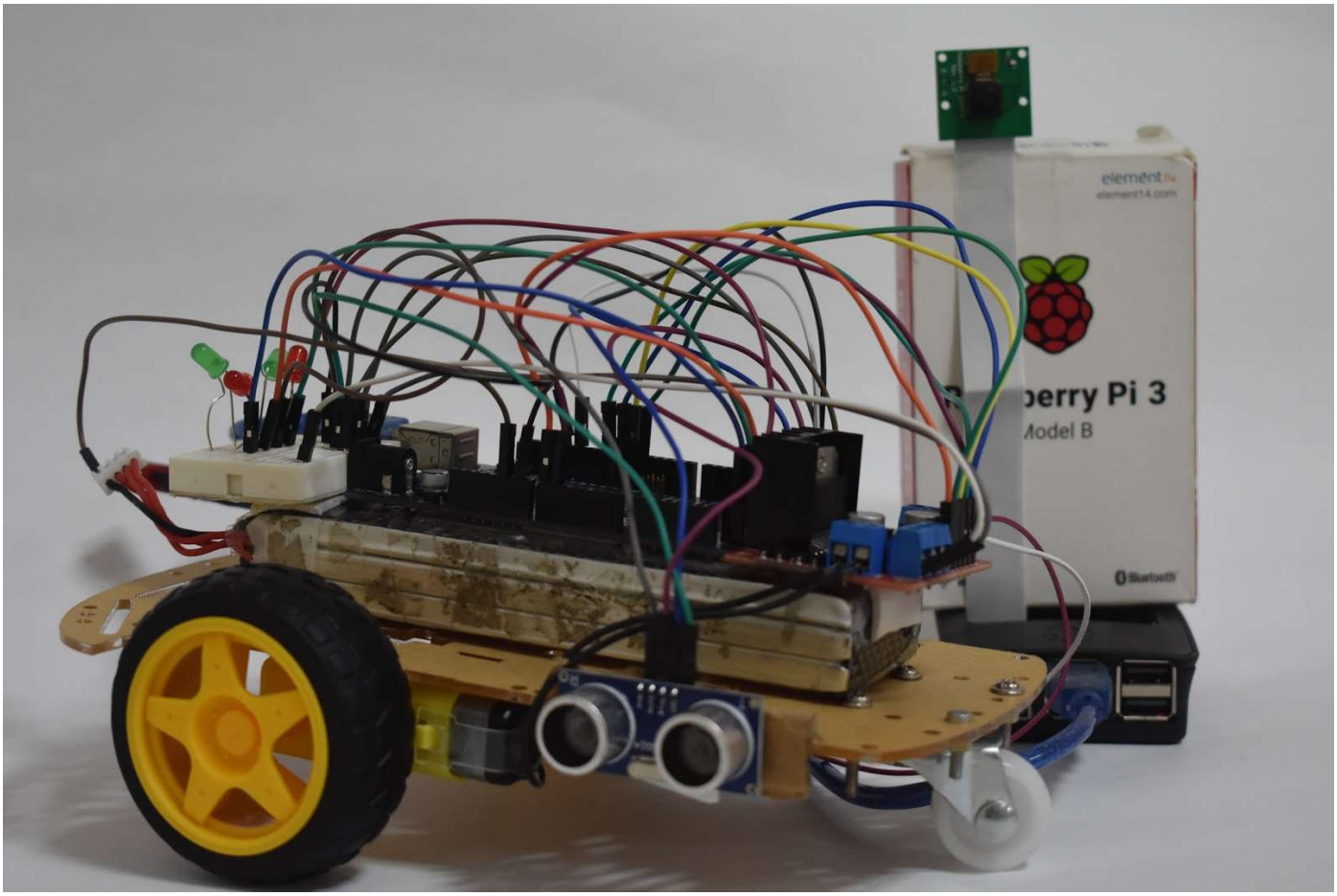
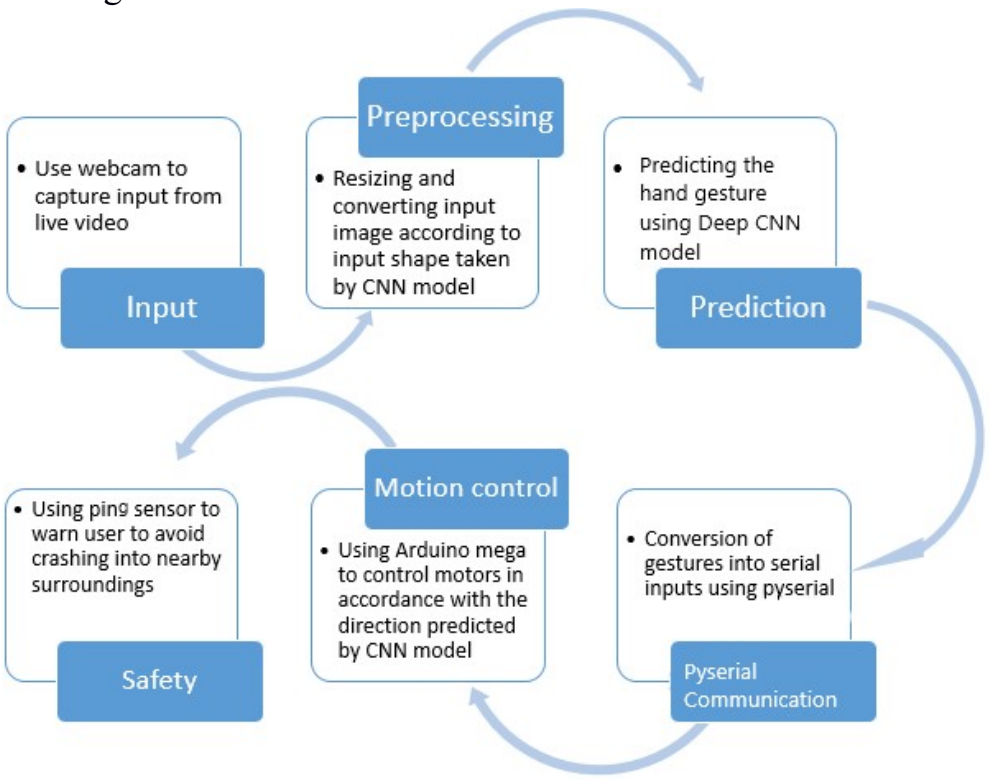
Mouvement De Geste



Introduction with Functional Block Diagram

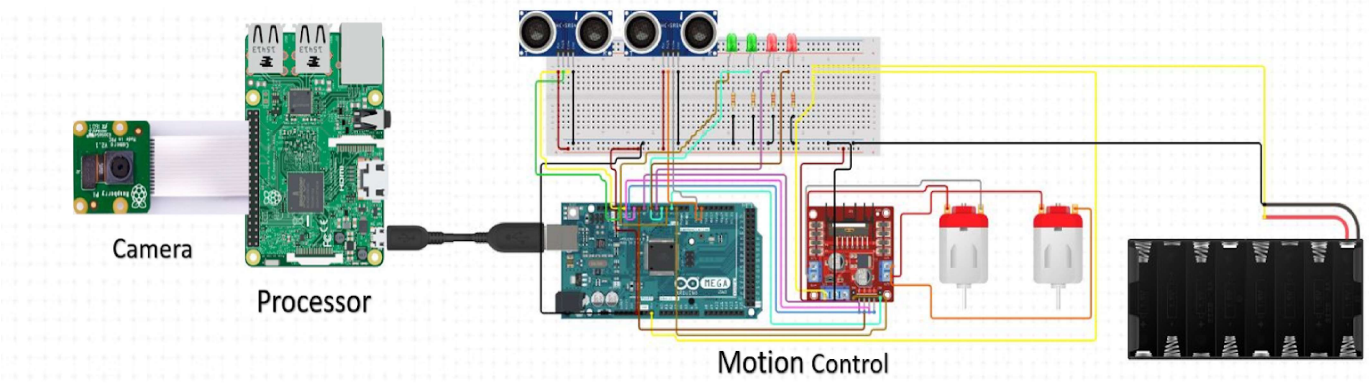
Aim: To develop a hand-gesture controlled maneuvering vehicle which would be useful to:
(i) Develop a system to control the self-driving car in its auto-pilot training phase.
(ii) Develop a controller for Augmented Reality(AR) based games.

We have developed a human-machine interface which can detect human hand gesture to accordingly maneuver a differential driven robotic car. Our system uses a webcam to continuously capture different frames from the live video stream. Our system relies on Deep Convolutional Neural Network(CNN) multiclass classifier to classify the captured frames into different hand gestures. The CNN model is trained on a self-collected dataset of 40,000 images comprising of 10,000 images of each class. The system then generates and transmits serial communication signals specific to each class to Arduino to control the car. The signal flow diagram is attached below:



Discussions on Design with Illustrations; Circuit Connections

Circuit Connections



Our project design includes two part:
(i) Capturing input
(ii) Preprocessing of captured images
(iii) Predicting the gesture in the captured frame
(iv) Generating and communicating the serial signals to the microcontroller(i.e. Arduino)
(v) Controlling the car motion
(vi) Feedback from the safety system

1. Capturing input :
we are using a webcam to capture frames from the live stream. The frames will consist of images of four different gestures namely left, right, stop and forward. the captured images are colored that is they contain all the three channels of RGB.

2. Pre-processing:
The shape of captured frames is(480, 640, 3), whereas input shape required by the model is (240, 240, 1). Therefore we resize our input image into the required shape. The resized colored image is then converted into a grayscale format. Conversion into grayscale is done so that the hand is detected properly and it is not clubbed with the background having similar RGB values. This also helps us in detecting bright objects independent of color, and colorful objects less dependent on brightness.

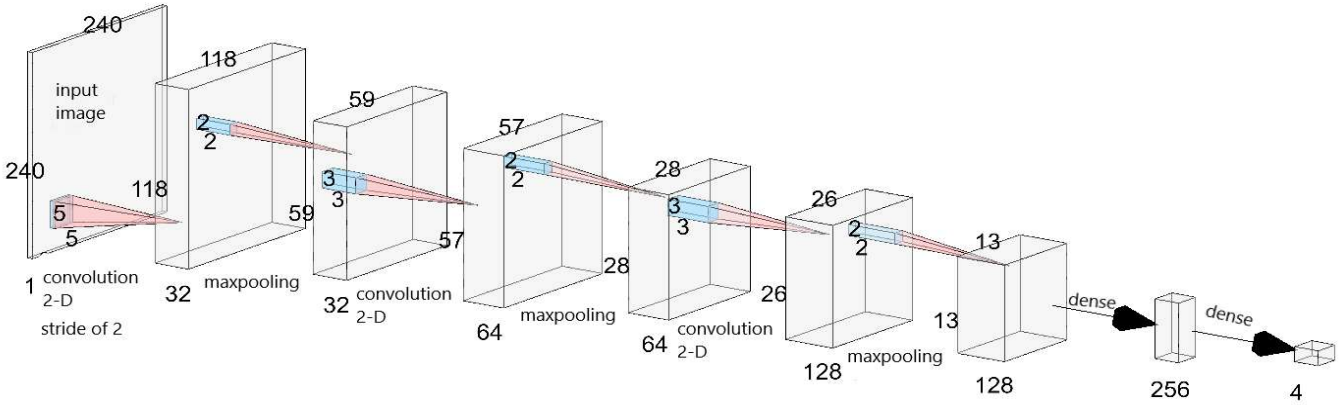
3. Predicting the gesture:
We predict the hand gesture by sending the preprocessed captured frame through a trained Deep Convolutional Neural Network multiclass classifier. We designed our classifier through the following steps:

(i) Dataset - We generated our own dataset comprising of total 40,000 images, i.e. 10,000 images for each class namely ‘forward’, ‘hold’, ‘left’ and ‘right’. The dataset was captured by a python code saving a frame per 100 milliseconds from live video stream via webcam. We captured the frames in different light settings and with all the possible meaningful positions of hand to improve the model performance. The dataset was then randomly divided into training and test set in 80:20 ratio for each class.

(ii) Data Augmentation - As our dataset was not big enough to get us good test accuracy, so we decided to perform augmentation on our dataset before training to improve the model’s generalization properties. The augmentation task included random zoom and rescaling.

(iii) Architecture - We have designed a Deep Convolutional Neural Network(CNN) consisting of 8 layers namely - 3 2D Convolutional layers with embedded ReLu activation function, 3 Max Pooling layers and 2 Fully Connected(FC) layers as shown in the figure below. The practical benefit of CNNs is that having fewer parameters greatly improves the time it takes to learn as well as reduces the amount of data required to train the model.
The reason behind using ReLu activation function over most common sigmoid function is that it trains much faster than the latter because the derivative of sigmoid becomes very small in the saturating region and therefore the updates to the weights almost vanish.

Another problem that this architecture solved was reducing the over-fitting by using a Dropout layer after the FC layer. Due to the dropout layer, different sets of neurons which are switched off, represent a different architecture and all these different architectures are trained in parallel with weight given to each subset and the summation of weights being one. This provides a structured model regularization which helps in avoiding the over-fitting.



4. Generating and Communicating Serial Signals:

We are using PySerial library of python to convert the processed data into serial data to communicate to Arduino Mega via Arduino USB Cable at a baud rate of 9600 bits per second. Once a particular gesture is detected by CNN Classifier we are assigning a unique value to each class and converting it into serial input using the following command line:
serial.write(b'0') //for forward gesture

5. Motion Control

(i) Arduino - We have coded the Arduino in such a way that each different serial input is mapped to specific action responsible for the smooth motion of the robotic car (say detection of left gesture will trigger the motors on right to turn left). We can control the motion of each wheel translationally as well as rotationally by changing the code properly.

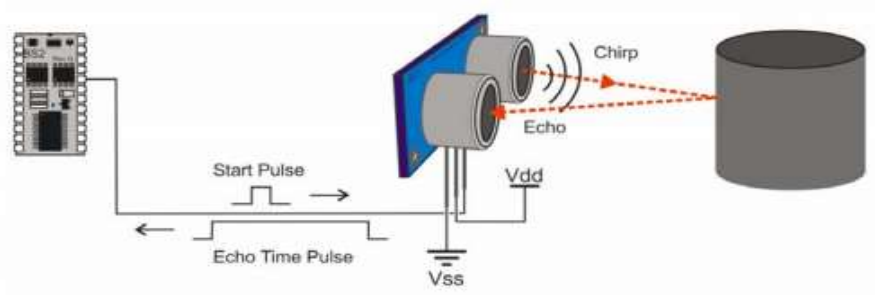
```
Initaized the variables
function void setup()
{
    // Initialize serial port to send and receive at 9600 baud
}
function void loop()
{
  // Check to see if at least one character is available
  {
    read the incoming input
    incoming data=c, c=0||1||2||3
   if(distanceL<=15 && distanceR<=15){
      turn on the left led and right red led
    }
   else if(distanceL<=15 && distanceR>15){
      turn on left red led and right green led
    }
   else if(distanceR<=15 && distanceL>15){
      turn right red led and left green led
    }
   else{
      turn left green led and right green led
    }
    switch (c) {
      case '0':move foreward
      case '1':hold
      case '2':left
      case '3':right
      default:hold

        break;
      }
  }
}
```

(ii) L298N Motor Driver-Motor Driver is used to control the speed and direction of rotation of both DC motors connected to right and left wheel of the car. The motor driver is powered by a 12V battery.

6. Safety Mechanism

We have used two ping sensors on the right and the left side of the car to identify the distance of surrounding obstacles. It detects the distance of the closest object in front of the sensor (from 3 cm up to 400 cm). It works by sending out a burst of ultrasound and listening for the echo when it bounces off of an object. It pings the obstacles with ultrasound. The Arduino board sends a short pulse to trigger the detection, then listens for a pulse on the same pin using the pulseIn() function. The duration of this second pulse is equal to the time taken by the ultrasound to travel to the object and back to the sensor. Using the speed of sound, this time can be converted to distance.



This sensor helps us to identify the potential threat of our car crashing into an object and vice versa. If the distance of any object in the surrounding of the car is less than a threshold value, then we alert the user by lighting a led on side of potential danger.

Experiments and Results; Performance Analysis:

(i) Experimenting with input method:

We are capturing the input frames via a webcam. To separate the hand shape from the environment, we thought of using masking or filtering of a definite color, but human skin color is spread over a wide spectrum of BGR(Blue Green Red) colorspace. In addition, there are always a lot of objects in our surrounding with similar color and shades as compared to human skin which makes it even harder. In our first attempt, we tried to separate the hand from the environment by wearing a dark purple colored hand glove because it is one of the rarest colors in our surroundings.

In our second attempt, we resorted to a completely opposite approach of separating the environment from hand. We are using a piece of black chart paper as the background of the human hand. This also helped us achieve our aim of making this project completely indigenous and gave us good accuracy on almost all the models compared to the former input method.

(ii) Experimenting with Architecture:

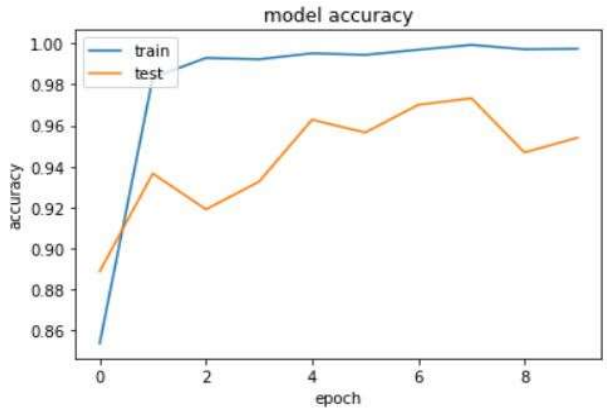
We started up building the architecture by testing with the number of Conv 2D layers. Initially, we started with 1 Conv2D layer, then moved on to 2 Conv2D layers, and finally settled with 3 Conv2D layers which gave the best performance on both validation set and even live video stream. We are training the model with a batch size of 32 over 10 epochs.

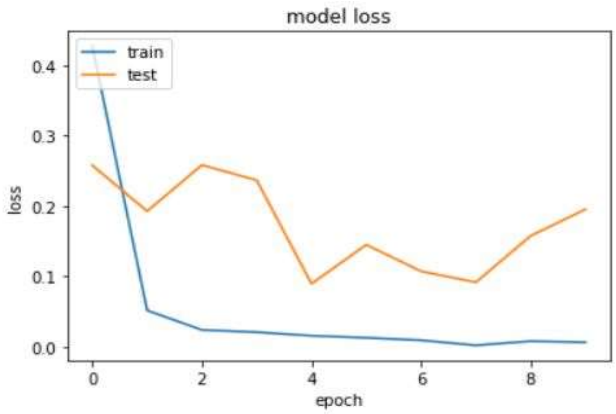
Results and Performance:

(A) 1 Conv2D layers - Model performed extremely good on the training set with average training accuracy over 10 epochs around 97% but performed significantly poor on the validation set with average accuracy over 10 epochs about 80%.This gap in numbers marks the overfitting of the model over the training dataset, so we needed to introduce a dropout layer with rate parameter as 0.5. The large number of parameters nearly 28 million parameters were also a reason for overfitting, thus we increased the number of Conv2D layers to decrease the number of parameters.

(B) 2 Conv2D layers - Model performed well on the training set as well as the validation set. The average training accuracy over 10 epochs was near 97% and average validation accuracy over 10 epochs was around 94% was good enough but it performed really poor on the live video stream, therefore, we increased one more Conv2D layer to learn even more complex features in the dataset.

(C) 3 Conv2D layers - Model performed better on the training set as well as the validation set. The average training accuracy over 10 epochs was near 98% and average validation accuracy over 10 epochs was around 94.5%. This model also performed excellent on the live video stream, therefore we decided to work with 3 Conv2D layer architecture. The model's performance can be analyzed through the model accuracy and model loss curve profiles shown below:





Summary and Future Work

We have successfully developed a system which can be used to capture and classify the hand gesture from a live video stream and establish serial communication with a microcontroller to do the specific task. We have designed an 8 layered Deep Convolutional Neural Network(CNN) multiclass classifier with an amazing test set accuracy of around 97% and validation set accuracy of around 94%. The system classifies each captured frame into any of four predefined classes namely - 'forward', 'hold', 'left' and 'right' and transmits a specific serial signal for each class through Arduino USB cable. Arduino Mega then generates signals at certain pins to generate the motion corresponding to the predicted class. Our system delivers excellent performance with a black background behind hand posing for different gestures. We believe that we are only a few steps away from achieving our aim and we would like to discuss the future work that can be done based on this project.

Future Work - We tried our best to make a perfect system but due to some of the hardware and software limitations during our project, our system lacks in few aspects which can be improved in future attempts.
(i) The precision of dc motor motion can be increased by replacing them with stepper motors.
(ii) Our system is currently wired. We can make it wireless by using Raspberry-Pi 3 as the processor and make use of its in-built WiFi module to transmit the data wirelessly to any microcontroller.
(iii) Our current system performs well in the presence of black background only. This can be solved in future work by collecting a sufficiently large and diverse dataset for different hand gestures with different backgrounds.
(iv) We can increase the number of hand gesture by collecting the appropriate data for each gesture.

List of Components

#	Item Name	Qty.	Provided by (Dept/Self/Guide)	Price (Rs.)
1.	Robotic Car Chassis & 2 DC Motors	1	Dept	
2.	Small-sized Breadboard(170 points)	1	Dept	
3.	Ultrasonic Distance Sensor	2	Dept	
4.	Jumper Wires Set	1	Dept	
5.	Lippo Battery 12V	1	Self	
6.	LED	4	Self	
7.	Arduino Mega	1	Self	
8.	Raspberry Pi 3	1	Guide	
9.	Pi Camera	1	Dept	
10.	Arduino USB Cable	1	Self	
11.	L298N Motor Driver	1	Self	