I.J. Intelligent Systems and Applications, **2024, 3, 59-73**
Published Online on June 8, 2024 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2024.03.05

A Self-driving Car Controller Module Based on Rule Base Reasoner

Anik Kumar Saha
Islamic University, Kushtia, 7003, Bangladesh
E-mail: anikjoy1997@gmail.com
ORCID iD: https://orcid.org/0009-0006-8783-050X

Md. Abdur Razzaque*
Islamic University, Kushtia, 7003, Bangladesh
E-mail: arazzaq@eee.iu.ac.bd
ORCID iD: https://orcid.org/0009-0009-1740-0784
*Corresponding author

Received: 26 December 2023; Revised: 02 January 2024; Accepted: 05 February 2024; Published: 08 June 2024

Abstract: The rapid improvement of sensing and recognition technology has had an impact on the vehicle sector that led to the development of self-driving cars. Thus, vehicles are capable of driving themselves without human interaction, mostly relying on cameras, various sensors technology, and advanced algorithms for navigation. In this research, a controller module of a self-driving car project is proposed using a rule baser reasoner that is capable to drive a car considering the health condition of the driver, the road lanes, traffic signs, obstacles created by other vehicles. A number of sensors including Global Positioning System module, camera, compass, ultrasonic sensor, physiological sensor (heartbeat, blood pressure, body temperature, etc.) are involved while reasoning. The proposed controller consists of several modules: sensor module, lane detection module, road sign and human detection module, reasoning module, Instruction execution module. According to the experimental results the proposed system is able to make correct decisions with a success rate of about 90-95%.

Index Terms: Controller, Computer Vision, Haversine Formula, Magnetic Sensor Calibration, Haar Feature-based Cascade Classifiers, Rule Based Reasoner.

1. Introduction

In the realm of transportation and technology, the longstanding dream of self-driving cars, also known as autonomous vehicles, is on the horizon. These autonomous vehicles hold the promise of revolutionizing transportation in multiple ways. It can alleviate traffic congestion and offer mobility solutions to assist the driver. One of the most significant benefits of self-driving cars is the potential for a substantial reduction in traffic accidents. Human error remains a leading cause of accidents, and this is where self-driving cars come into play [1,2]. By relying on sensors and advanced algorithms, it can swiftly and accurately make decisions, thus lowering the risk of accidents resulting from distractions, impaired driving, or poor decision-making. Notably, recent advancements in both hardware and software have accelerate the building of self-driving car.

According to "AN OVERVIEW OF DEVELOPMENT GPS NAVIGATION FOR AUTONOMOUS CAR" paper, the robot car is navigated based on GPS data and compass information. In this setup, the GPS receiver provides location coordinates, while the compass supplies the heading value of the robot car to the Controller. Using this data, the controller autonomously determines the driving direction of the car and guides it towards the designated goal point. This research presents a prototype of a GPS-based car that follows no specific path planning, only location coordinates are known. Based on these coordinates, the car moves straight forward to the destination. There is no function for road lane tracking. Radhakrishnan et al. used a camera to capture the traffic sign and through image processing techniques, the traffic sign is detected in real-time. Road lanes are detected using the HSV method with the camera in [3-5]. After lane detection, the controller instructs to drive the car in the appropriate direction. Hence, the car continuously drives by detecting the road lane until the road lane finishes.

From the above recent works, it is assumed that further research work is necessary to develop a self-driving car that can drive in real environment smoothly. In this research work we have proposed a prototype of a self-driving car controller with embedded platform (Raspberry Pi, Arduino). This prototype controller is able to reach the destination by detecting the road lane. Additionally, it can detect traffic signs and, based on the information from these signs, it moves in the proper direction, comparing it with compass data. Another feature that was added is that if the car faces any accidents or passengers need any medical emergency assistance, it sends an SMS to relatives containing the current location of the car.

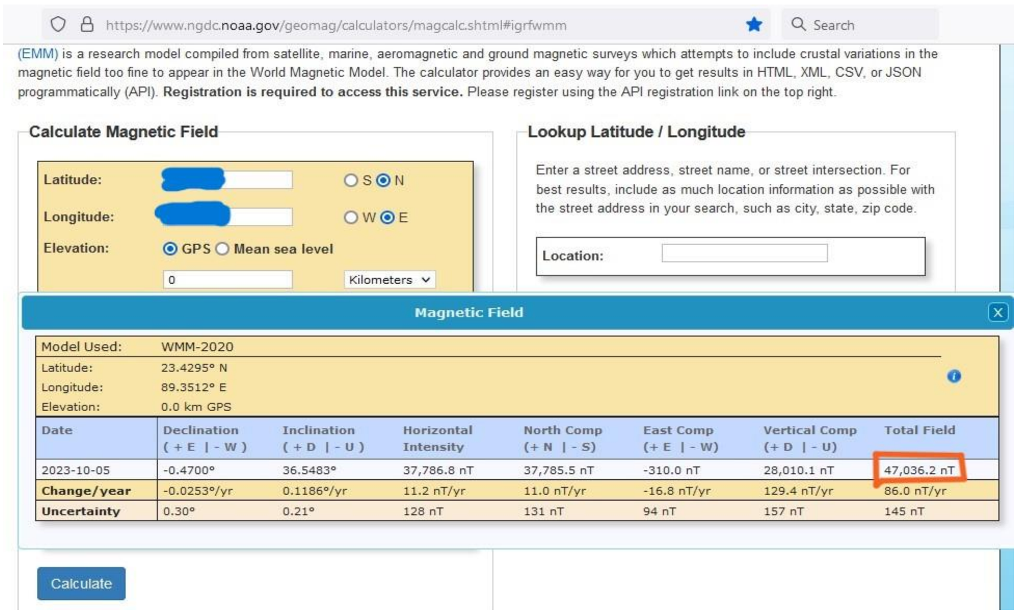
In this proposed system, a camera and a compass sensor are used to drive. Despite the active utilization of Lidar and Radar systems in the advancement of perception technologies, camera and compass offer a cost-effective and potent means of extracting environmental information. This system is also able to send text massage to relative for helping the wounded people, if the car faces any accident.

The proposed self-driving car controller that consists of several modules: sensor module, lane detection module, road sign and human detection module, reasoning module, Instruction execution module. Firstly, it continuously measures distance and direction using ultrasonic sensor, GPS location, and compass. Using the Haversine formula, the sensor module can easily calculate the distance between the goal point and the current location, as well as the car's direction. Detects obstacles in front of the vehicle to avoid collisions. Check the driver's physiological data using physiological sensors. Secondly, it features road lane detection. The camera continuously captures images and employs image processing techniques such as gray scaling, image thresholding, and perspective transformation to detect road lanes. Thirdly, it includes road signs and human detection using the Haar cascade algorithm. Fourthly, based on road lane detection, recognition of traffic signs, and human detection, the controller generates instructions regarding the car's direction in reasoning module. If the driver's pulse rate deviates from the normal range due to an accident or any health issue, Arduino immediately sends an SMS with the location to close relatives to save lives.

Overall, the primary goal is to create an affordable controller for a self-driving car that can efficiently make decisions in busy traffic, alert about the driver's health issues, and supervise a low-skilled driver.

2. Proposed Method

A self-driving car must be capable of detecting obstacles in front of it to avoid accidents. Typically, Radar sensors are used for obstacle detection. In the event of an accident in a sparsely populated area where a person or passengers may lose consciousness, it can be challenging to rescue them. However, this self-driving car project offers a valuable solution for the rescue of injured individuals. The satellite based Global Positioning System (GPS) is used to set the destination location for this research. GPS enables the precise determination of location, time, velocity and the tracking of objects or people [6]. GPS receivers have the capability to provide latitude and longitude coordinates with high accuracy, typically within an error margin of about 2.5 to 5 meters in normal conditions, such as when there is good sky visibility and a high number of visible satellites.


https://www.ngdc.noaa.gov/geomag/calculators/magcalc.shtml#igrfwmm

(EMM) is a research model compiled from satellite, marine, aeromagnetic and ground magnetic surveys which attempts to include crustal variations in the magnetic field too fine to appear in the World Magnetic Model. The calculator provides an easy way for you to get results in HTML, XML, CSV, or JSON programmatically (API). **Registration is required to access this service.** Please register using the API registration link on the top right.

Calculate Magnetic Field

Latitude: [redacted] S [✓ N]

Longitude: [redacted] W [✓ E]

Elevation: [✓ GPS] Mean sea level

0 Kilometers

Lookup Latitude / Longitude

Enter a street address, street name, or street intersection. For best results, include as much location information as possible with the street address in your search, such as city, state, zip code.

Location:

Magnetic Field

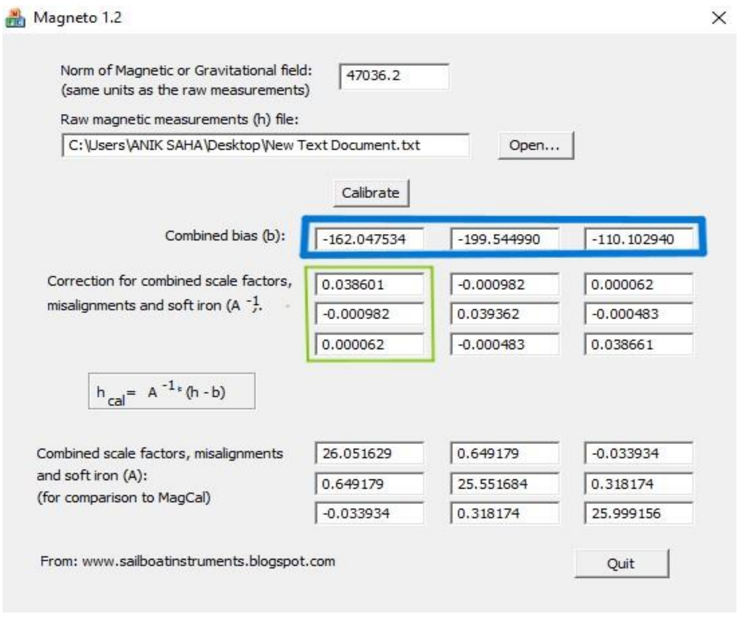
Model Used:	WMM-2020
Latitude:	23.4295° N
Longitude:	89.3512° E
Elevation:	0.0 km GPS

Date	Declination (+ E | - W)	Inclination (+ D | - U)	Horizontal Intensity	North Comp (+ N | - S)	East Comp (+ E | - W)	Vertical Comp (+ D | - U)	Total Field
2023-10-05	-0.4700°	36.5483°	37,786.8 nT	37,785.5 nT	-310.0 nT	28,010.1 nT	47,036.2 nT
Change/year	-0.0253°/yr	0.1186°/yr	11.2 nT/yr	11.0 nT/yr	-16.8 nT/yr	129.4 nT/yr	86.0 nT/yr
Uncertainty	0.30°	0.21°	128 nT	131 nT	94 nT	157 nT	145 nT

Fig.1. Magnetic field value

2.1. Calibration of Compass

The triple-axis magnetic sensor is used to identify the car's direction/angle (heading value). Initially the direction is not correct as shown in Fig. 3. So, it is necessary to calibrate the sensor. For calibrating the sensor, we use the Magneto 1.2 software. The magnetic field norm values obtaining from the NOAA website [7] put into the Magneto 1.2 software to get the combined bias value and combined scale factors as shown in Fig. 1 and Fig. 2. The combined bias value and combined scale factors are then used to get the calibrated values according to algorithm 1 that gives the correct heading value for the car. The correct calibrated values are shown in Fig. 4. This compass sensor must be positioned at least 20-25 cm away from other metallic components to avoid interference. During the car's operation, this sensor must be kept at a certain distance from iron and magnets to ensure that its direction value is not affected by them.



Magneto 1.2

Norm of Magnetic or Gravitational field: (same units as the raw measurements): 47036.2

Raw magnetic measurements (h) file: C:\Users\ANIK SAHA\Desktop\New Text Document.txt

Combined bias (b): -162.047534 | -199.544990 | -110.102940

Correction for combined scale factors, misalignments and soft iron (A^{-1}):

0.038601	-0.000982	0.000062
-0.000982	0.039362	-0.000483
0.000062	-0.000483	0.038661

$h_{cal} = A^{-1} * (h - b)$

Combined scale factors, misalignments and soft iron (A): (for comparison to MagCal)

26.051629	0.649179	-0.033934
0.649179	25.551684	0.318174
-0.033934	0.318174	25.999156

From: www.sailboatinstruments.blogspot.com

Fig.2. Combined bias value and combined scale factor in magneto 1.2 software

Table 1. The algorithm of triple-axis magnetic sensor calibration

Algorithm 1 Compass Calibration	
	Input: Reading Compass values along three axis **x**, **y**, **z**, and bias_x, bias_y, bias_z, scale_fact_x1, scale_fact_x2, scale_fact_x3, scale_fact_y1, scale_fact_y2, scale_fact_y3, scale_fact_z1, scale_fact_z2, scale_fact_z3, **Output:** Calibration value x_Cal, y_Cal, z_Cal
1.	x_offset=x-bias_x
2.	y_offset=y-bias_y
3.	z_offset=z-bias_z
4.	x_Cal= scale_fact_x1*x_offset+ scale_fact_x2*y_offset+ scale_fact_x3*z_offset
5.	y_Cal= scale_fact_y1*x_offset+ scale_fact_y2*y_offset+ scale_fact_y3*z_offset
6.	z_Cal= scale_fact_z1*x_offset+ scale_fact_z2* y_offset+ scale_fact_z3*z_offset
	return x_Cal, y_Cal, z_Cal

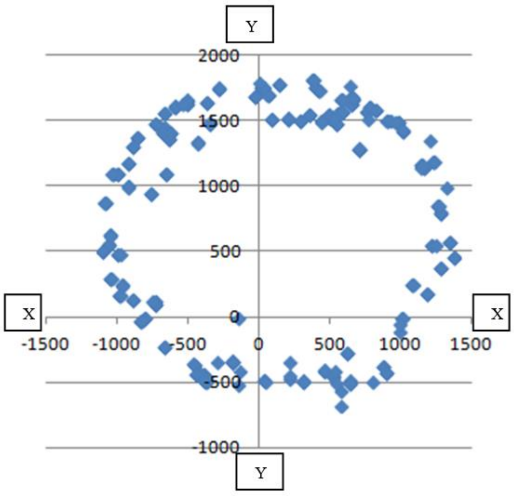


Fig.3. Raw magnetometer values (before calibration)

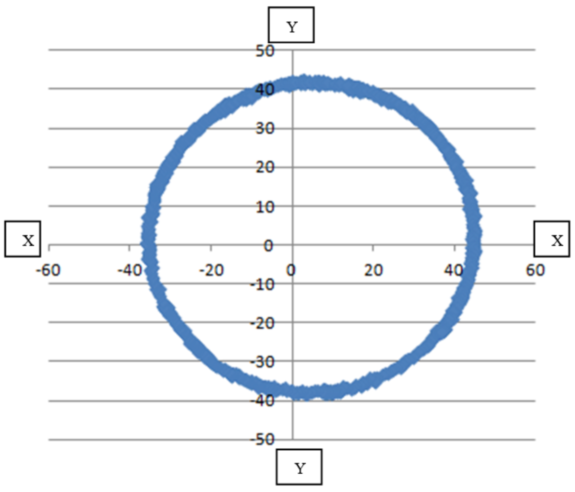


Fig.4. Raw magnetometer values (after calibration)

2.2. Distance and Waypoint Angle Measurement

Generally, for measuring the distance between two points, the Haversine formula [8] is more familiar. The Haversine formula allows the calculation of the great-circle distance between two points, which is the shortest path over the Earth's surface. We used the Haversine formula to measure the distance from the current point to the destination point.

$$a = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat_1) \times \cos(lat_2) \times \sin^2\left(\frac{\Delta lon}{2}\right) \tag{1}$$

$$c = 2 \times a\tan 2\left(\sqrt{a}, \sqrt{1-a}\right) \tag{2}$$

$$d = R \times c \tag{3}$$

where, lat_1 = latitude of point_1, lon_1 = longitude of point_1, lat_2 = latitude of point_2, lon_2= longitude of point_2, d = distance between two points, R = radius of earth, Δlat = lat_2- lat_1lat2 - lat1, Δlon = lon_2- lon_1.

The waypoint angle is the angle measured clockwise from the north to the destination, with north serving as the reference point and considered as 0° which is shown in Fig. 4. This waypoint angle can be determined using the forward Azimuth formula, which also calculates the angle between the car's heading and the destination. The car heading value is obtained from the compass. By subtracting this heading value from waypoint angle required turn of the car is obtained to reach goal point.

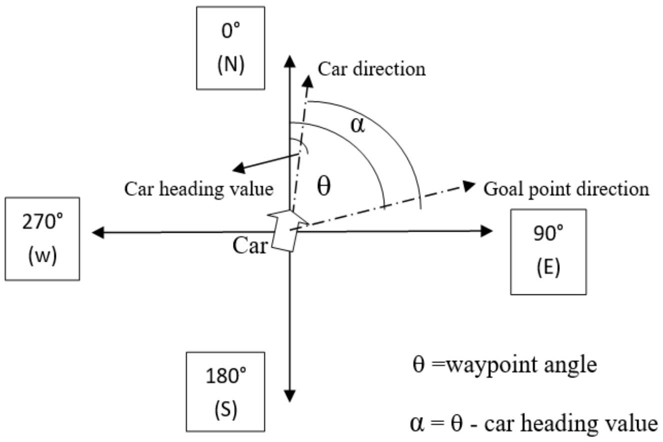


Fig.5. Distance and waypoint angle measurement

$$e = \cos(lat_2) \times \sin(\Delta lat) \tag{4}$$

$$f = \left(\cos\left(lat_1\right)\times\sin\left(lat_2\right)\right)-\left(\sin\left(lat_1\right)\times\cos\left(lat_2\right)\times\cos\left(\Delta lon\right)\right) \tag{5}$$

$$\theta = a\tan 2\left(e, f\right) \tag{6}$$

where lat_1 = latitude of point_1, lon_1 = longitude of point_1, lat_2 = latitude of point_2, lon_2 = longitude of point_2, $\Delta lat = lat_2 - lat_1$; $\Delta lon = lon_2 - lon_1$, θ = waypoint angle.

2.3. Road Lane Detection

The captured images of the road track and sends to the controller to detects the road lane. This captured image processing is carried out using the open-source computer vision (OpenCV) library, that offers a wide range of image processing functions, including gray scaling, image thresholding, perspective transformation, and more [9]. Firstly, the RGB image are converted into grayscale image then again converted into a binary image by comparing each pixel's value with a threshold value. If the threshold value is higher than the pixel value, it is set to the minimum value; otherwise, it is set to the maximum value. In this project, a threshold of 127 is assigned as the minimum value, while 255 is assigned as the maximum value. The perspective transformation is associated with changes in the point of view. This proposed technique concentrates only road lane [10]. In perspective transformation, four points are required in the input image, as well as four corresponding points in the output image. The cv2.getPerspectiveTransform() function is used, followed by the cv2.warpPerspective() function, to obtain a 2-dimensional view of the left road lane. Both the original image and the perspective-transformed image are depicted in Fig. 6 & Fig. 7 .

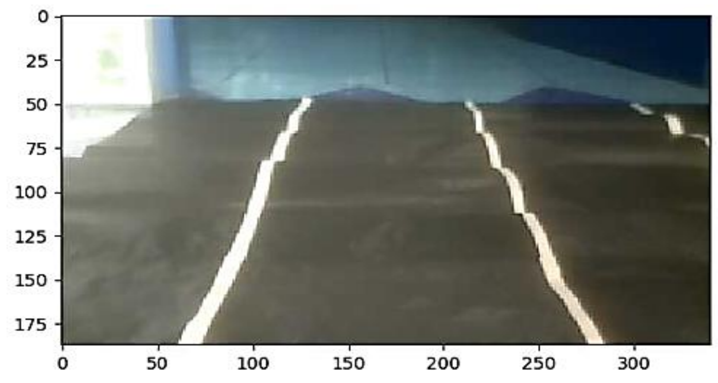


Fig.6. Perspective transform (Input)

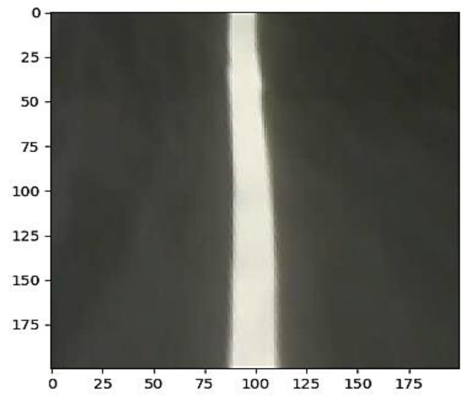


Fig.7. Perspective transform (Output)

2.4. Road Line Marking

After performing the perspective transform, a fixed area of the left-side road lane is obtained. The cv2.moments() function is then used to draw a straight white line from the center of the road lane to the car's heading point, while another straight green line was drawn to indicate the center of the car's direction. By comparing both lines, the controller provides instructions to the actuator to steer the car in the correct direction. Fig. 8-10 shows the car navigation direction.

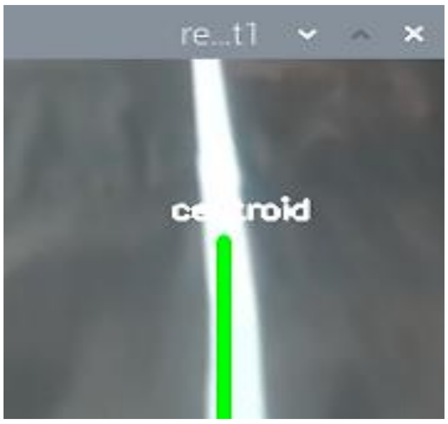


Fig.8. Car navigation directions (Forward)

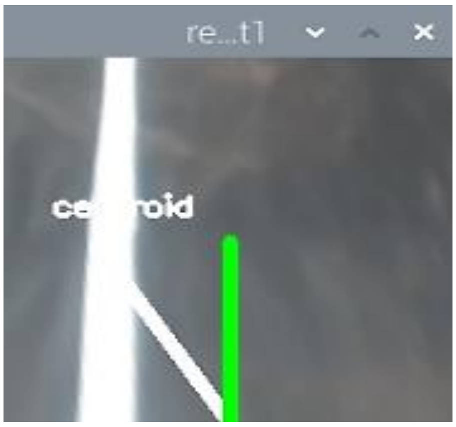


Fig.9. Car navigation directions (Left)

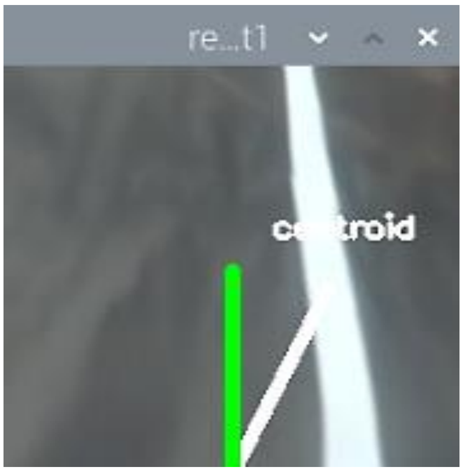


Fig.10. Car navigation directions (Right)

2.5. Traffic Sign and Person Detection and Recognition

Detecting traffic signs is crucial for guiding the car in the right direction. In this research, we focus on various types of traffic signs, including crossroads, left or right turn, right or forward, left or forward, and stop signs. During the operation of the self-driving car prototype, it makes decisions about the driving direction based on GPS data when it detects a traffic sign. If a human is detected in front of the car, it comes to a stop.

Traffic signs and humans are detected using an object detection method known as Haar feature-based cascade classifiers [11]. The Haar feature is a feature based on wavelets, utilizing a mathematical function with a rectangular shape for image decomposition. The term "Haar" specifically denotes a mathematical function with a rectangle-shaped waveform. The training of the cascade function involves using numerous negative and positive images [12]. In this context, positive images represent the objects that need to be detected. For example, in the case of detecting the crossroad sign, only images of crossroad signs are considered as positive image, while all other images are used in this project, such as background images, road images, person images, left or right-side sign images, etc., are considered negative images.

To train the cascade function, a sample folder is created, which contains two subfolders: one for positive images and another for negative images. This sample folder is used for training through the 'Cascade Trainer GUI,' which produces the cascade.xml file. This XML file is then used to detect specific objects [12].

The images of traffic signs and people are continuously captured by the Pi camera in real-time, which are then converted into grayscale images. The cascade.xml file is uploaded through cv2.CascadeClassifier(). The detectMultiScale() function is used to detect objects of different sizes in the grayscale image and provides x, y coordinates, height, and width of the detected objects. Then, using these values, rectangular shapes are drawn around the detected objects. This process enables the detection of both traffic signs and people, as illustrated in Fig. 11-16.

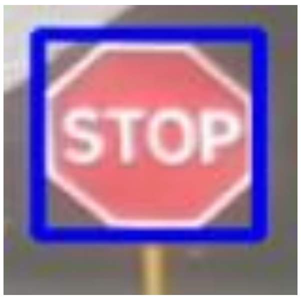


Fig.11. Stop sign

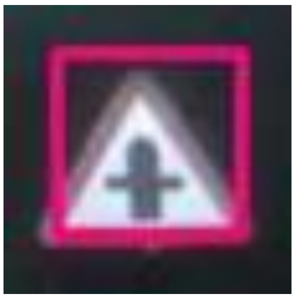


Fig.12. Crossroad sign

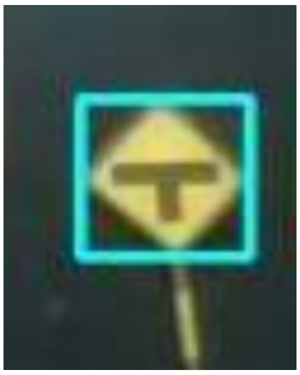


Fig.13. Right or left sign

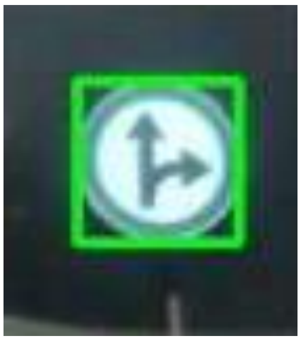


Fig.14. Forward & right sign

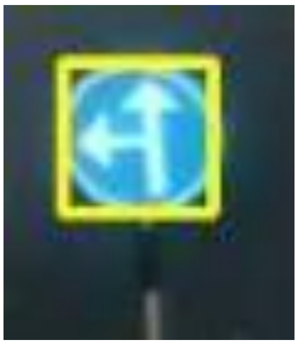
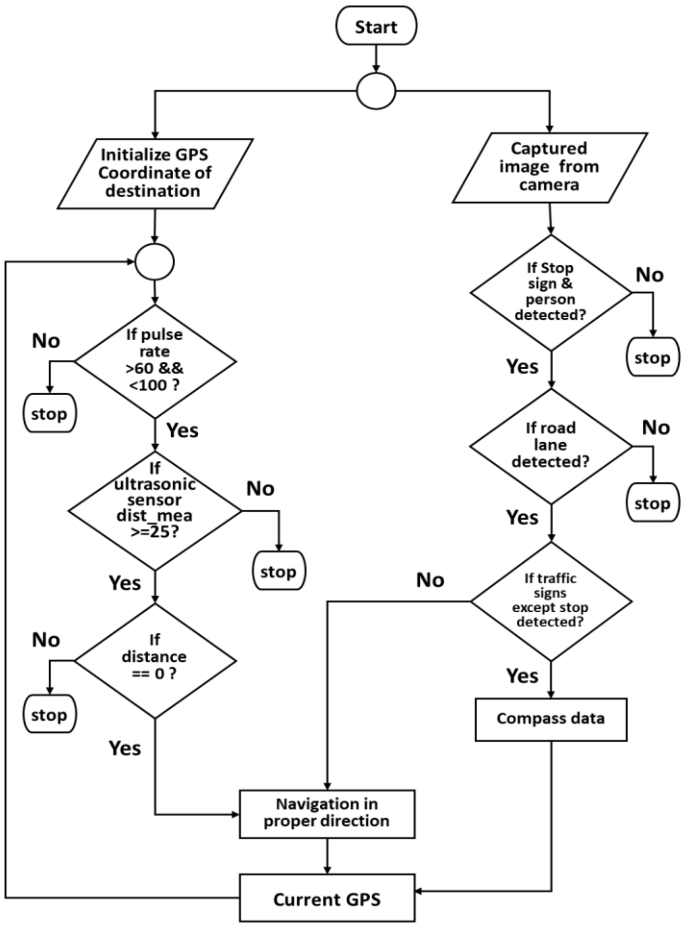


Fig.15. Forward & left sign

Fig.16. Person

2.6. Rule based Reasoner

A rule-based reasoner is designed and implemented in the controller to generate the instructions according to the different context collected from different types of sensor data. Some of the rules are as follows:


Start
Initialize GPS Coordinate of destination
Captured image from camera
If pulse rate >60 && <100 ?
No
stop
Yes
If ultrasonic sensor dist_mea >=25?
No
stop
Yes
If distance == 0 ?
No
stop
Yes
If Stop sign & person detected?
No
stop
Yes
If road lane detected?
No
stop
Yes
If traffic signs except stop detected?
No
Yes
Compass data
Navigation in proper direction
Current GPS
Fig.17. Flowchart of the controller

- When the "forward or right" sign is detected, and the distance between the car and that traffic sign is less than 50 cm, and the angle difference between the goal point and the car's heading value obtained from the compass is less than or equal to 30 degrees, the car proceeds straight forward and then proceed forward based on detected road lane. Otherwise, the car rotates 90 degrees to the right based on GPS data and then proceeds forward based on detected road lane.
- When the "forward or left" sign is detected, and the distance between the car and that traffic sign is less than 50 cm, and the angle difference between the goal point and the car's heading value obtained from the compass is less than or equal to 30 degree, the car proceeds forward and then proceeds straight forward based on detected road lane. Otherwise, the car rotates 90 degree to the left based on GPS data and then proceeds forward based on detected road lane.
- When the "crossroad" sign is detected, and the distance between the car and that traffic sign is less than 50 cm, and the angle difference between the goal point and the car's heading value obtained from the compass is less than or equal to 20 degree, the car proceeds straight forward and then proceeds forward based on detected road lane. Otherwise, the car rotates 90 degree to the left or right based on GPS data and then proceeds forward based on detected road lane.
- When the "left or right" sign is detected, and the distance between the car and that traffic sign is less than 50 cm, and the angle difference between the goal point and the car's heading value obtained from the compass is greater than or equal to 20 degree, the car rotates 90 degree to the left or right based on GPS data and then proceeds forward based on detected road lane. Otherwise, the car comes to a stop.

The flowchart of the reasoner is shown in Fig. 17.

3. Prototype of the Rule-based Controller

The car is equipped with a GPS module and a compass, which are used to reach the destination. A Pi camera is used to analyze road lanes, humans, and traffic signs such as stop, crossroad, forward or right, forward or left, and left or right. An ultrasonic sensor is used to detect any obstacles, such as cars. Additionally, a GSM module and a pulse sensor are used to detect the health condition of passengers in case the car faces an accident, or a passenger needs medical emergency assistance. The whole system consists of microcontroller such as Arduino Mega that act as an actuator control module and Raspberry pi4 as a rule-based controller, Pi-camera, Ublox Neo 6M GPS module, QMC5883L compass module, L298n motor driver, HC-05 Bluetooth module, bidirectional logic shifter, pulse sensor, Ultrasonic sensor HC-SR04 [13-15], SIM800L module, Optical encoder, DC power supply and other supporting equipment. A pulse sensor is employed to continuously monitor a person's heart rate and send the data to the Arduino Mega. In addition, we have used some software Integrated development environment (IDE) i.e. Arduino IDE, VNC viewer for Raspberry Pi. Arduino IDE is more familiar for programming software which is available for Windows, macOS and Linux. A built-in library manager is included into it. As a result, it is easy to add and manage libraries. It provides a code editor where one can write, edit and save Arduino code. Arduino code is based on C/C++. And VNC stands for Virtual Network Computing which is available for Windows, macOS and Linux. VNC Viewer is a client side software. In this project, it is used to control the Raspberry pi. The overall view of a self-driving car is shown in Fig. 18.

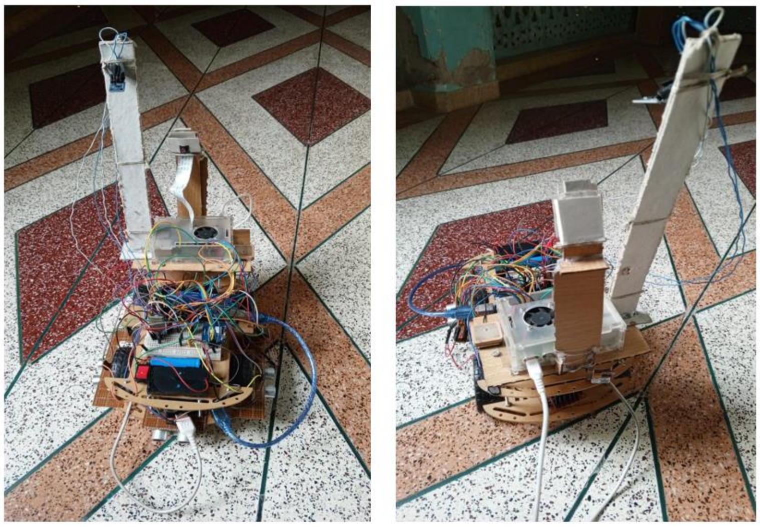


Fig.18. Self-driving car prototype

4. Result and Discussion

This project experiment is done outside below the sky. The overview of the road track is shown in Fig. 19. The size of the road track is about 20 m. The distance between the goal point and starting point is 17.60 m. During the experiment, the car stopped about 50 to 70 cm from the goal point due to low accuracy of GPS receiver.

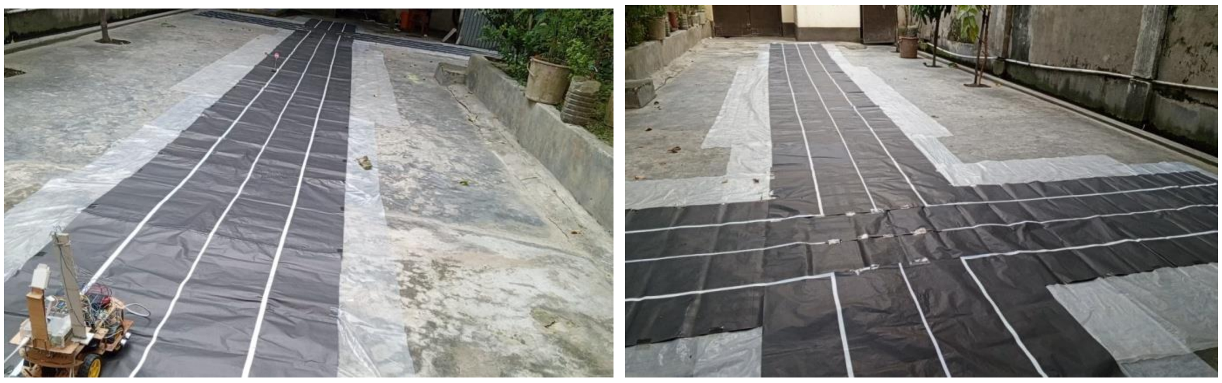


Fig.19. Road track for evaluation

4.1. Collision Analysis and Stop Sign & Person Detection Result

This experiment was conducted with a non-moving toy car. As the proposed system moved toward its destination, it automatically stopped at a 20 cm distance behind the toy car. The threshold is designed in such a way that the self-driving car prototype stops upon detecting any obstacle within a 30 cm distance. However, due to excessive speed, it came to a halt after covering an additional 10 cm towards the toy car. When the ‘Stop Sign’ & ‘Person’ were detected, the car immediately stopped. Fig. 20 shows the different stop cases.

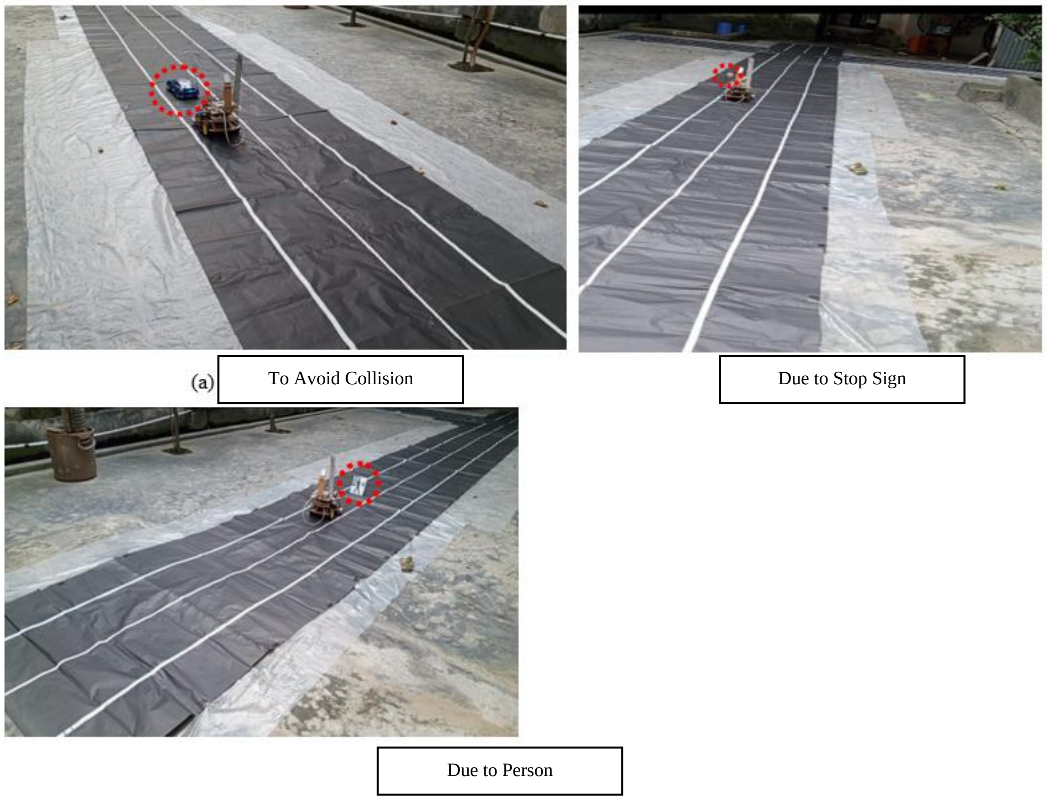


Fig.20. Self-driving car stop cases

4.2. Pulse Test Result

When a passenger's heart rate deviates from the normal range 60 to 100, either dropping too low or rising too high, which can occur due to a car accident or physical illness and is detected through a pulse rate sensor, the Arduino Mega sends an SMS containing the car's location coordinates through the SIM800L module to the designated relatives, as previously described. During this experiment, we started to breathe heavily putting the index finger on the pulse sensor. At that moment, the pulse rate immediately reached 105 and the Arduino immediately disconnected the power supply to the motor driver. Then, an SMS containing the location coordinates was sent to the provided number.

4.3. Traffic Sign and Human Detection Result

When traffic signs, such as 'crossroad', 'forward or right', 'forward or left' and 'left or right' are detected through the Pi Camera, the Raspberry Pi provided information to the Arduino Mega based on the corresponding traffic signs which are already mentioned in section 2. When traffic signs, excluding the stop sign, are detected, the optical encoder is activated. It is employed as a counter to determine the car's position at the road's turning point. If the car didn't reach its destination, the Arduino Mega made decisions about the car's direction based on the information from the Raspberry Pi, compass data, and GPS data. When the Pi camera detected a 'stop sign' as well as 'person', Arduino stopped the car.

In this experiment, we provide two destination coordinates for two observations, respectively, for every traffic sign testing. Hence, the car's directions for every traffic sign are different. For the crossroad, in the first observation, the car turned left by 90 degrees and then proceeded along the road lane. In the second observation, the car moved straight and then continued along the road lane, as depicted in Fig. 21.

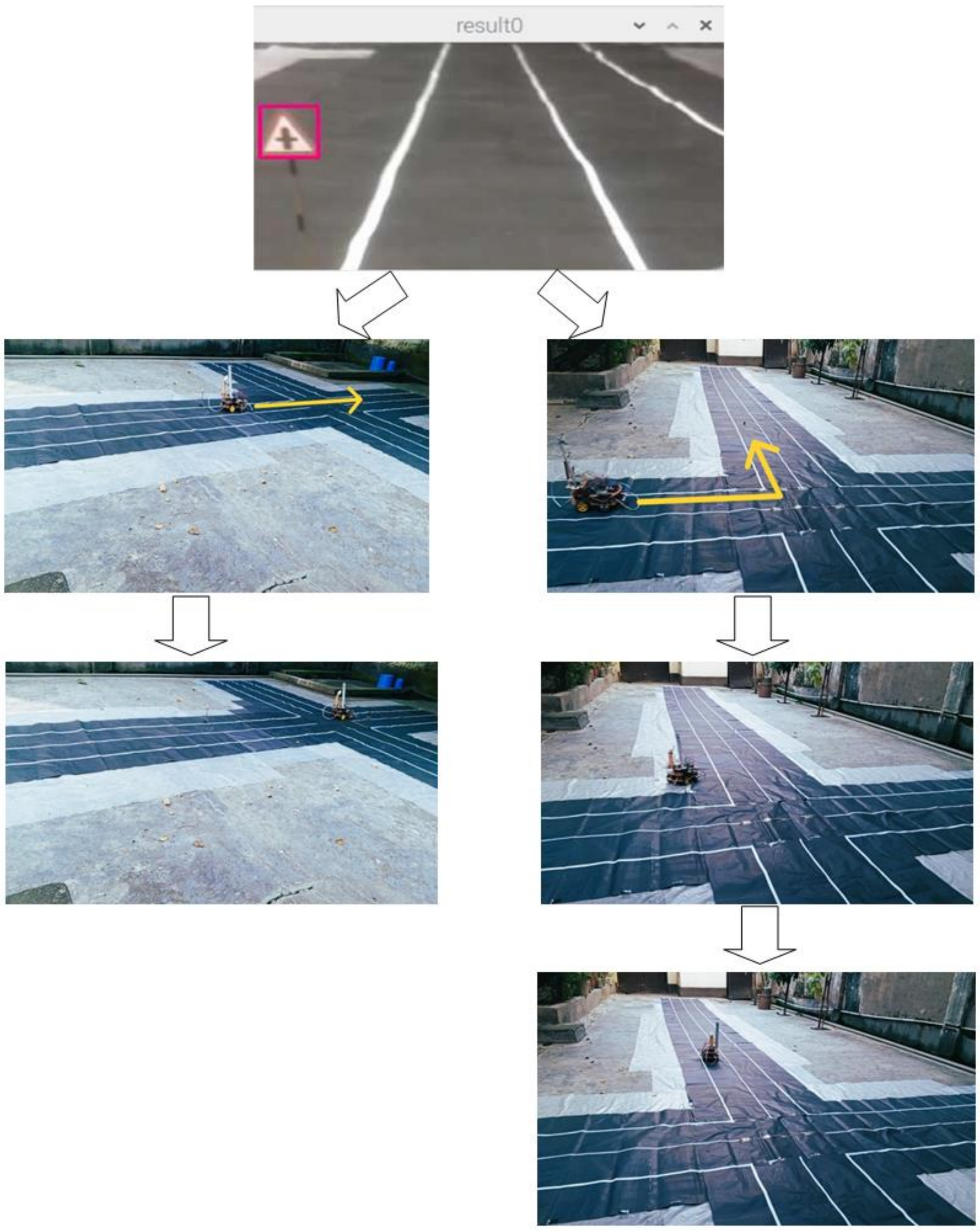


Fig.21. Car direction detection based on crossroad sign and compass data

For the forward or right traffic sign, the car may rotate in the right by 90 degree and proceeded along the road lane or the car moves straight and then proceeds straight forward based on detected road lane, as previously mentioned. In the first observation, the car rotated 90 degrees to the right based on GPS data and then proceeded along to the detected road lane since the difference was greater than 20 degrees. In the second observation, the car moved straight and then continued along the detected road lane because the difference was less than 20 degrees.

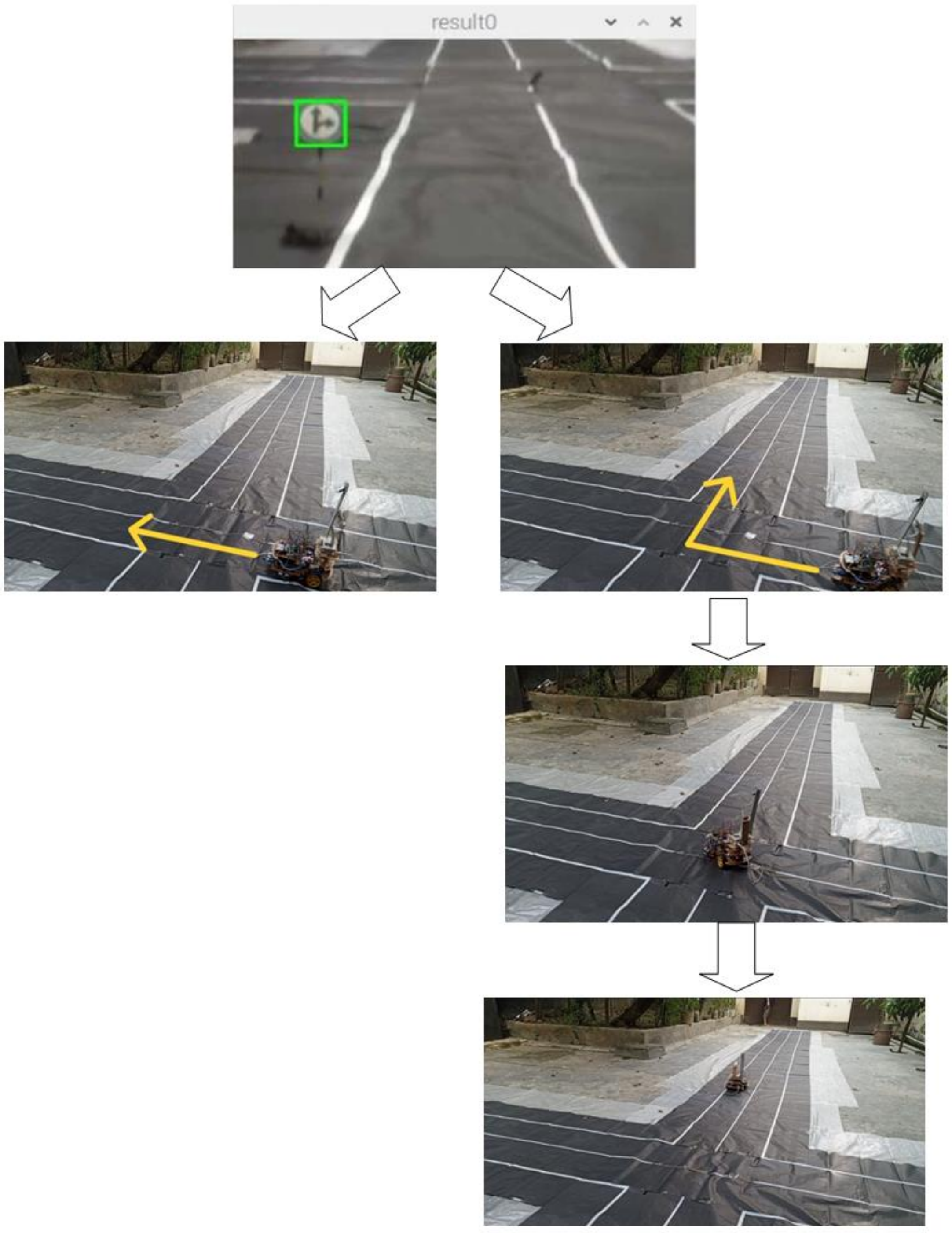


Fig.22. Car direction based on forward or right sign and compass data

The prototype of the self-driving car exhibited a proper response according to the forward or left sign, as clearly depicted in Fig. 23.

For a left or right traffic sign, the car may rotate in the right or left direction by 90 degree or come to a stop, as previously mentioned. In the first observation, the car rotated 90 degree to the left based on GPS data and then proceeded along the detected road lane since the difference was greater than 20 degree. In the second observation, the car stopped because the difference was less than 20 degree. The main reason for stopping the car was to allow the driver or passenger to select the direction.

4.4. Limitation

Despite the higher accuracy, this proposed system is not so efficient to drive in multiple road lanes. The road lane detection algorithm largely relies on pre-processing to get desire result. As well as the car goes outside from the definite road lane during 90 degrees turning. Except these problems, the whole system is able to take 90-95% correct decision.

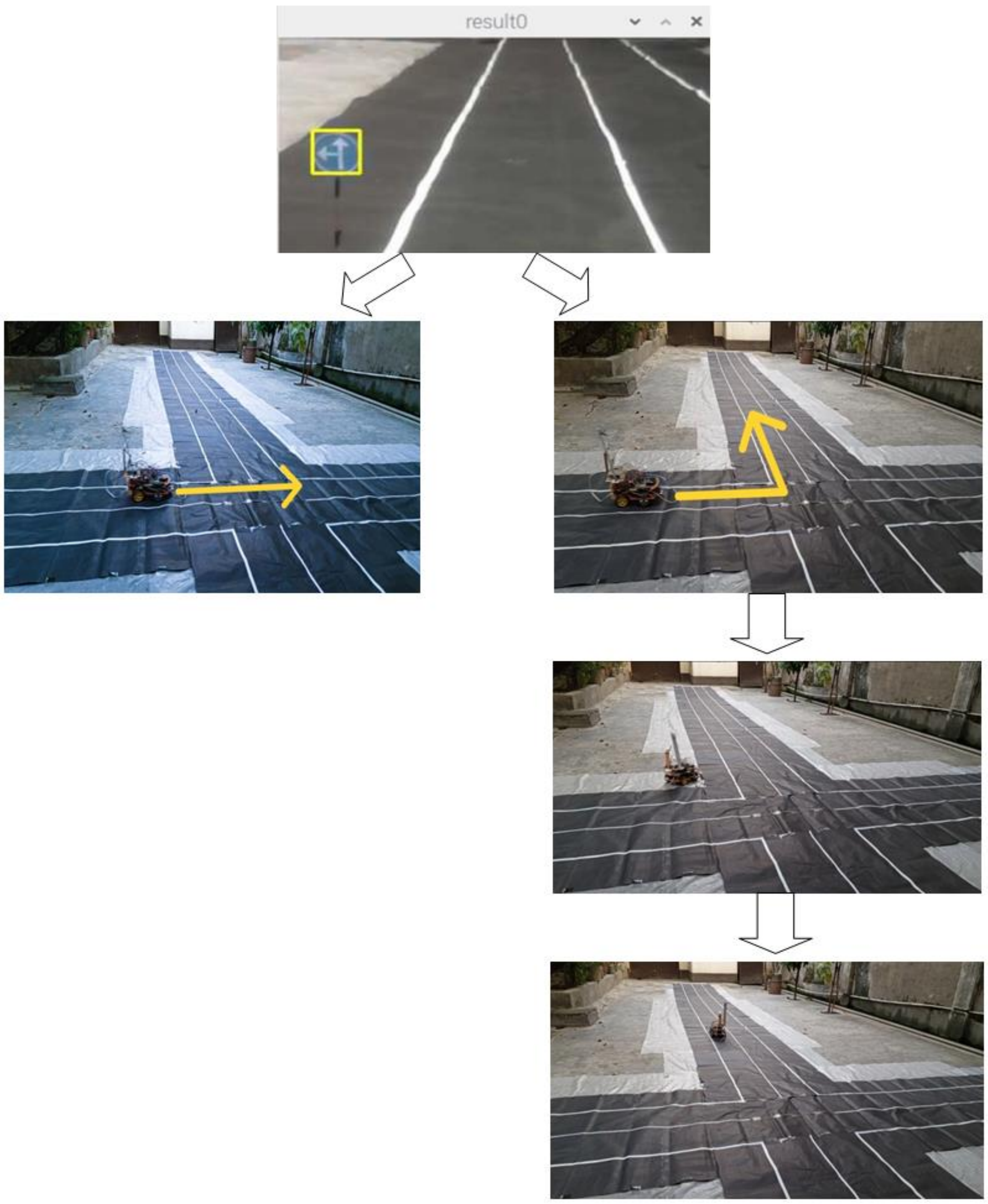


Fig.23. Car direction based on forward or left sign and compass data

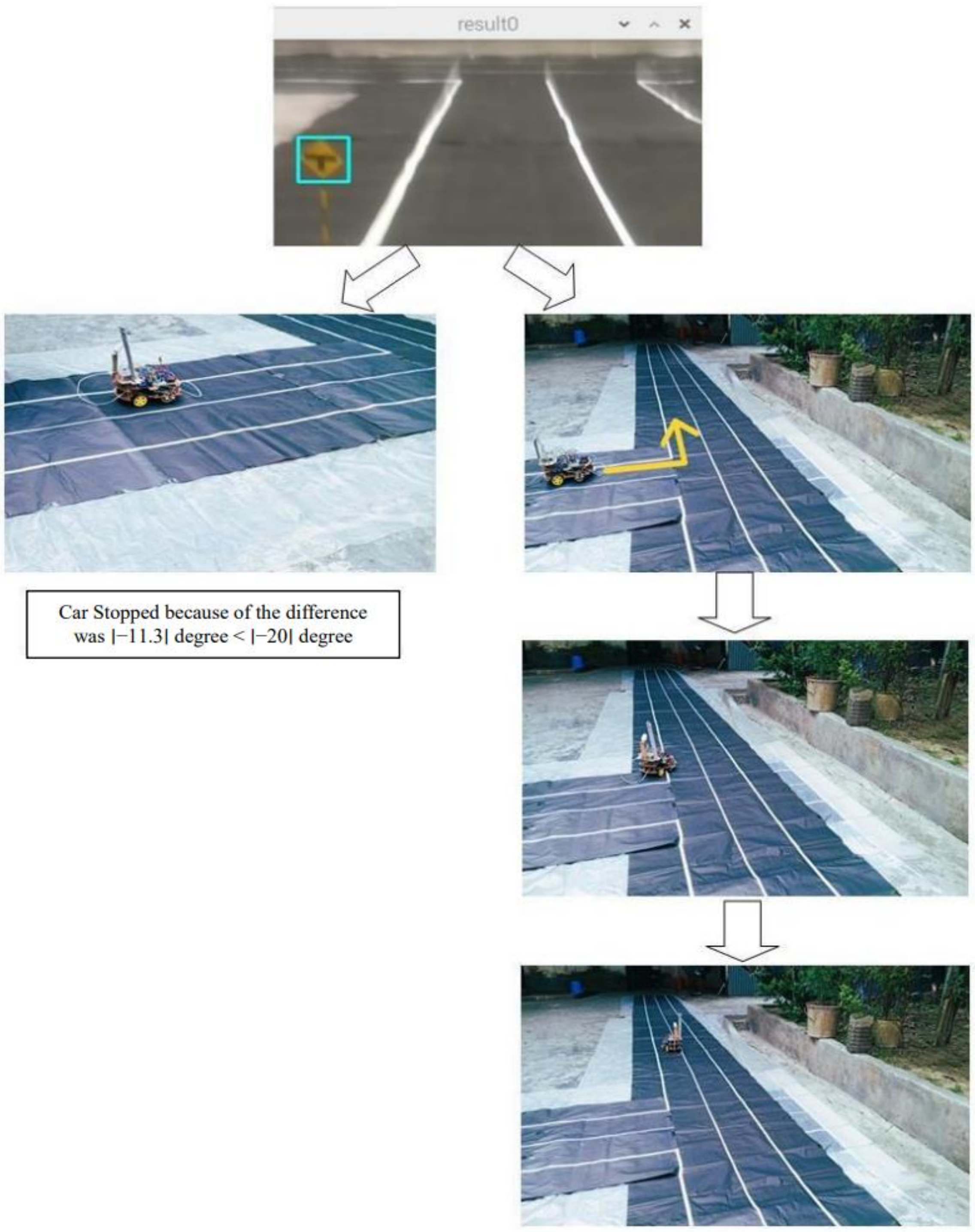

result0
Car Stopped because of the difference was |−11.3| degree < |−20| degree

Fig.24. Car direction based on left or right sign and compass data

5. Conclusions & Future Work

We have proposed a rule-based controller that aims to provide more safety, reliability and drive itself. It is able to take accurate decision to reach goal point as well as detect traffic sign, person & obstacle successfully. This system is also able to take proper decision according to the driver's health condition. To recapitulate, the proposed self driving car is better compared to conventional techniques. For getting more accurate decision, there is a plan to use a high megapixel camera for getting better image as well as make more efficient to drive in multiple road lane.

6. Code of the Proposed System

https://github.com/AnikKumarSaha7/A-Self-Driving-Car-Controller-Module-Based-on-Rule-Base-Reasoner.git

References

[1] N. Wong, C. Chambers, K. Stol, and R. Halkyard, “Autonomous Vehicle Following Using a Robotic Driver”, Institute of Electrical and Electronics Engineers (IEEE), pp. 115-120, 2009. DOI: 10.1109/MMVIP.2008.4749517.
[2] M. Fathy, N. Ashraf, O. Ismail, S. Fouad, L. Shaheen, and A. Hamdy, “Design and implementation of self-driving car”, Procedia Computer Science, Vol. 175, pp. 165-172, 2020. DOI: 10.1016/j.procs.2020.07.026.
[3] W. Rahiman and Z. Zainal, “An Overview of Development GPS Navigation for Autonomous Car”, Institute of Electrical and Electronics Engineers (IEEE), pp. 1112-1118, 2009, DOI: 10.1109/ICIEA.2013.6566533.
[4] N Radhakrishnan, S Maruthi, “Real-Time Indian Traffic Sign Detection Using Raspberry Pi and Open CV”, International Journal of Advance Research in Science and Engineering (IJARSE), pp. 1488-1499, 2017.
[5] K. K. Roestamadji, F. B. Setiawan, L. H. Pratomo, and S. Riyadi, “Implementation of Self Driving Car System with HSV Filter Method”, Journal RESTI (Rekayasa Sistem dan Teknologi Informasi), Vol. 7, No. 3, pp. 430-435, 2023. DOI: 10.29207/resti.v7i3.4579.
[6] D. C. R Ranasinghe, W. K. I. L. Wanniarachchi, and U .A .D . N. Anuradha, “GPS guided auto sensing system for motor vehicles”, Institute of Electrical and Electronics Engineers (IEEE), pp. 184-188, 2019. DOI: 10.23919/SCSE.2019.8842715.
[7] Edmallon, “How to calibrate a compass (and accelerometer) with Arduino. Retrieved from The Cave Pearl Project: https://thecavepearlproject.org/2015/05/22/calibrating-any-compass-or-accelerometer-for-arduino/”, 2015.
[8] Irtsam Ghazi, Ihtisham ul Haq, Muhammad Rashid Maqbool, and Sanaan Saud, “GPS Based Autonomous Vehicle Navigation and Control system”, Institute of Electrical and Electronics Engineers (IEEE), pp. 238-244, 2016. DOI: 10.1109/IBCAST.2016.7429883.
[9] Yasin Yeniaydin, Klaus Werner Schmidt, “A Lane Detection Algorithm Based on Reliable Lane Markings”, Institute of Electrical and Electronics Engineers (IEEE), 2018. DOI: 10.1109/SIU.2018.8404486.
[10] Raja Muthalagu, Anudeepsekhar Bolimera, V. Kalaichelvi, “Lane detection technique based on perspective transformation and histogram analysis for self-driving cars”, Computers & Electrical Engineering, Vol. 85, pp. 1-16, 2020. DOI: 10.1016/j.compeleceng.2020.106653.
[11] Diajeng Tyas Purwa Hapsari; Cindykia Gusti Berliana; Putri Winda; M. Arief Soeleman, “Face Detection Using Haar Cascade in Difference Illumination”, Institute of Electrical and Electronics Engineers (IEEE), pp. 556, 2018. DOI: 10.1109/ISEMANTIC.2018.8549752.
[12] Adrian Rosebrock, “ OpenCV Haar Cascades. Retrieved from pyimagesearch: https://pyimagesearch.com/2021/04/12/opencv-haar-cascades/”, 2021.
[13] Raspberry Pi Arduino Serial Communication, “https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/”, (n.d.).
[14] M. Haritha, T. Kavitha, G. Bhavadharni, and V. Prabhu, “GPS Based Autonomous Vehicle Navigation In Robotics Along With Directionality”, International Journal of Pure and Applied Mathematics, Vol. 119, No. 15, pp. 1603-1608, 2018.
[15] A. Ravoor, S. S. Pawar, and V. S Indrali, “Development of Pulse Rate Indicator in Real Time”, International Journal of Engineering Research & Technology (IJERT), pp. 226, 2020.

Authors’ Profiles

Anik Kumar Saha received the B.Sc. engineering in Electrical and Electronic Engineering from the Islamic University, Kushtia 7003, Bangladesh, in 2023. His research interests are Autonomous Vehicles and Machine Learning and IoT.

Md. Abdur Razzaque received the B.Sc. and M.Sc. degrees from Islamic University, Kushtia, Bangladesh, in 2001 and 2003. He is a Faculty Member at the Department of Electrical and Electronic Engineering, Islamic University, Kushtia 7003, Bangladesh. His research interests include Signal Processing, Machine Learning, and Artificial Intelligence.

How to cite this paper: Anik Kumar Saha, Md. Abdur Razzaque, "A Self-driving Car Controller Module Based on Rule Base Reasoner", International Journal of Intelligent Systems and Applications(IJISA), Vol.16, No.3, pp.59-73, 2024. DOI:10.5815/ijisa.2024.03.05