Self Learning Car in Simulated Environment

Sai Prudhvi Valicherla
dept. of Computer Science
Georgia State University
Atlanta, Georgia, United States of America
svalicherla1@student.gsu.edu

***Abstract*—Vehicular technology is very competitive and demanding with Tesla's successful introduction of electric cars. A self-driving car can sense its surroundings and make decisions through traffic and other obstacles with minimum or no human interactions. The market for autonomous driving is booming, and building accurate positioning technologies to factor in different types of uncertainties, such as pedestrian behavior, random objects, and types of roads and their settings, is critical in achieving safe and reliable intelligent transportation systems. Such systems bring revolution in transportation for physically and mentally challenging individuals. This project aims to demonstrate the self-learning car in the simulated environment and is the first step toward autonomous cars. Considering the car's physics, laws of motion, the center of mass, rotation angle, traffic rules, road borders, sensors, collision detection, ray tracing, neural networks, visualization, and genetic algorithm, this project is a simple implementation of the self-learning car.**

***Index Terms*—Laws of motion, Sensors, Ray tracing, Collision detection, Neural Networks, Network Visualization, and Genetic Algorithm.**

I. Introduction

Although the future is uncertain, planning involves anticipating forthcoming circumstances and requirements. Many decision-makers and professionals are engaged in how autonomous vehicles, also known as self-driving or robotic vehicles, will affect future travel patterns and the need for public transportation, parking spaces, and roads. Researchers also want to know what public policies can reduce the risks and maximize the benefits of these vehicles. Optimistic projections frequently oversee significant obstacles and costs. Many technical issues must be resolved before self-driving cars can operate reliably in normal conditions. They will require years of testing and regulatory approval to progress from an idea to full commercial availability on the Technology Readiness Level scale, so they can become affordable and appealing to consumers. Motor vehicles are expensive, long-lasting, and heavily regulated, so new vehicle technologies typically take decades to enter fleets. Autonomous driving is a combination of unexpected problems; a camera, phone, or computer failure can be frustrating but rarely fatal, but motor vehicle system failures can be both frustrating and deadly to occupants and other road users. As a result, autonomous vehicles will most likely take longer to develop and provide a smaller net benefit.

Technological advancements in recent decades have made vehicles easier and safer to drive. Seatbelts, airbags, power steering, anti-lock braking systems, and automatic transmission are now standard features in most vehicles. Vehicles have recently included additional safety features such as reversing cameras and sensors for obstacles in the driver's blind spots. These features, commonly referred to as 'driver-assist,' provide an early taste of the potential benefits of developments in autonomous vehicle technology. Soon, autonomous vehicles, also known as self-driving or driverless vehicles, are expected to be capable of driving in most conditions without human intervention. Autonomous vehicles eliminate the human element of driving and promise to increase accessibility for those currently unable to drive, such as the young, elderly, and mobility impaired. Even for those who do not want or cannot afford to own a private vehicle, autonomous vehicles are likely to reduce the cost of taxis and ride-sharing services, benefiting a broader range of people and businesses. They are also likely to reduce parking demand, allowing some parking infrastructure to be repurposed to benefit people in other ways. Finally, because autonomous vehicles do not have the closed set of vision or reaction times as humans, they can communicate with one another and not only expected to improve vehicle safety but also increase road capacity by allowing vehicles to travel more closely together.

II. Literature Survey

Europe's Eureka Project PROMETHEUS [1], carried out between 1987 and 1995, was one of the first major automated driving studies. The project resulted in the development of VITA II by Daimler-Benz, which successfully automated highway driving [2]. The DARPA Grand Challenge, held in 2004 by the US Department of Defense, was the first major automated driving competition in which all participants failed to complete the 150-mile offroad parkour. The challenge was made difficult by the rule that no human intervention at any level was permitted during the finals. In 2005, a similar DARPA Grand Challenge was held. This time, five teams completed the off-road course without human intervention [3]. Many different research organizations from all over the world tested their ADSs in a test environment that was fashioned after an everyday urban scene during the DARPA Urban Challenge [4], held in 2007. Six teams completed the tournament. The test set did not have some aspects of a real-world urban driving scene, such as pedestrians and bicycles, even though this competition was the biggest and most important event up to that point. Nevertheless, the fact that six teams finished the task garnered much attention. Following the DARPA Urban Challenge, several other automated driving contests, including

[5], [6], [7], [8], were organized in other nations. Since the DARPA challenges, many self-driving car competitions and trials have been performed. Relevant examples include the European Land Robot Trial (ELROB) (Schneider & Wildermuth, 2011), which has been held from 2006 to the current year; the Intelligent Vehicle Future Challenge (Xin et al., 2014), from 2009 to 2013; the Autonomous Vehicle Competition, from 2009 to 2017 (SparkFun, 2018); the Hyundai Autonomous Challenge, in 2010 (Cerri et al., 2011); the VisLab Intercontinental Autonomous Challenge, in 2010 (Broggi et al., 2012); the Grand Cooperative Driving Challenge (GCDC) (Englund et al., 2016), in 2011 and 2016; and the Proud-Public Road Urban Driverless Car Test, in 2013 (Broggi et al., 2015). At the same time, research on self-driving cars has accelerated in academia and industry worldwide. Notable examples of universities researching self-driving cars comprise Stanford University, Carnegie Mellon University, MIT, Virginia Tech, FZI Research Center for Information Technology, and the University of Ulm. Notable examples of companies include Google, Uber, Baidu, Lyft, Aptiv, Tesla, Nvidia, Aurora, Zenuity, Daimler and Bosch, Argo AI, Renesas Autonomy, Almotive, AutoX, Mobileye, Ambarella, Pony.ai, Idriverplus, Toyota, Ford, Volvo, and Mercedes Benz. Most ADSs use a variety of sensors and algorithms on different modules and break the challenging work of automatic driving into smaller tasks. End-to-end driving has recently begun to take the place of modular techniques. In many of these tasks, deep learning models have taken the lead [9]. To define the prerequisites for the effective deployment of autonomous vehicles, [10] paper used six requirements. The primary criteria are fault tolerance, stringent latency, architecture, resource management, and security. In the research work [11], tools and datasets for autonomous driving are used to compare topics such as societal impact, system design, and object and image detection in a real-world context. The evolution, characteristics, and trends related to autonomous driving are identified in [12] research paper, using keyword analysis and their corresponding burst strengths. CiteSpace has been used to identify the border aspects with 96 fields. The implementations and design challenges are divided into sections in [13] research, and further sub-sections include cost, software complexity, creation of digital maps, simulation, and validations. Additionally, it examines privacy, decision-making, resource computation, and safety factors. [14] article examines the available base to comprehend the impact, discusses policy concerns, and reveals planned trajectories of potential gaps in the knowledge. It also stresses the importance of getting cities ready for driverless vehicles in conclusion. A growing number of businesses and organizations for scientific research have invested in this area. Self-driving car development has involved companies like Google, Tesla, Apple, Nissan, Audi, General Motors, BMW, Ford, Honda, Toyota, Mercedes, Nvidia, and Volkswagen [15]. Based on its strong foundation in artificial intelligence, Google is an Internet firm pioneer in self-driving automobiles [16]. Two Google self-driving cars were put through road tests in June 2015. The vehicles used by Google have completed more than 3.2 million kilometers of testing to date, making them the most practical. Tesla is a different business that has made significant strides in autonomous vehicles. Tesla was the first business to invest in self-driving technology for commercial use. Its "auto-pilot" technology has recently made significant advancements, followed by the Tesla model series. The National Highway Traffic Safety Administration (NHTSA) only classifies Tesla's autopilot technology as being at the Level 2 stage. However, as one of the companies that have successfully applied the autopilot system, Tesla demonstrates that the car has essentially achieved automatic driving under certain circumstances [17]. Numerous automobile and Internet businesses have recently turned their attention to the self-driving car market. Zenuity, a joint venture between Volvo and Autoliv in Sweden, is dedicated to ensuring the safety of autonomous vehicles [18]. The South Korean government permitted Samsung to test its autonomous vehicles on public roads in 2017. Samsung filed the most patent applications worldwide in self-driving automobiles from 2011 to 2017 [**?**]. In 2013, the Baidu deep learning institute oversaw China's self-driving car research project. Baidu founded the automobile networking business branch in 2014, and the company subsequently released CarLife, My-car, CoDriver, and other products [20].

III. Modules

- Car Mechanics
- Borders and Lanes
- Sensors
- Reading the borders
- Collision detection
- Traffic
- Neural network
- Network Visualization
- Genetic Algorithm

A. Car Mechanics

This module looks into designing the car, its position in the coordinate plane, its border, and control type, and defining the car's speed, acceleration, friction, color, and movements in the coordinate plane.

1) Position of the car:

- The parameter defines the placement of the car in the coordinate plane. It is represented by the points x, y, where x refers to the relative position on the x-axis and y represents the relative position on the y-axis.
- The position of the car refers to the point in the coordinate plane and will be at the center of the car's mass.

2) Width and Height:

- The Width represents the extension of the car on the x-axis.
- The car width is extended to the left and right of the car's center of mass.
- The Height represents the extension of the car on the y-axis.

- The car height is extended in the upward and downward directions of the car's center of mass.

3) Control type: This project considers three types of controls for the car.

- Keys - if the arrow keys on the keyboard control the vehicle or if controlled by the user.
- Traffic - If the default settings control the vehicle, movement with constant speed in the forward or the backward directions or if the car is resting on the road.
- AI - If the car is connected to the neural networks and makes its own decisions to evade the traffic or road borders.

4) Max Speed:

- The max speed refers to the car's maximum speed in the forward and reverse directions.
- The max speed of the car in the forward direction is set to 3, and in the reverse direction, it is -1.5.
- The product of the rotation angle of the car and speed in that direction defines the speed of the car in different directions.
- Acceleration of the car is the rate of change of velocity according to the time.
- The speed in the forward direction is defined as $speed = speed + acceleration$
- If the speed of the car is more than the max speed over time, then the speed of the car is set to the max speed of the car
- The speed in the backward direction is defined as $speed = speed - acceleration$
- If the car's speed is less than half of the negative value of the max speed, then the car's speed is set to negative of half of the max speed.
- The car's speed is bounded between the Max speed and negative time half of the max speed.

5) Friction: Friction is the grip between the road and the car, and this property helps to maintain fluid movement of the car in all directions.

- If the car's speed is greater than zero, then the friction comes into play and decreases the car's speed.
- $speed = speed - friction$
- If the car's speed is less than zero, then the friction comes into play and increases the car's speed.
- $speed = speed + friction$
- If the car's speed is in the range of 0 and friction value, then the car is set to rest.

6) Color:

- If the control type of the vehicle is traffic, then the color will be randomized.
- If the control type of the car is 'AI,', and if the car is not damaged, the car's color is blue, and if the car is damaged, then the color will be grey.

7) Controls:

- The movement of the car in different directions. Forward, Right, Reverse, Left.

8) Movement in different direction:

- The car is rotated by an angle of 0.03 upon clicking either the left or right directions.
- This angle is used to determine the speed of the car in the direction and the distance covered by the car in the direction.
- The direction of the car is determined using the below formula
- $(if speed! = 0)$
- $flip = speed > 0?1 : -1$
- $if(controls.left)angle+ = 0.03 * flip$
- $if(controls.right)angle- = 0.03 * flip$
- The distance traveled by car on the $x - axis$ is given by $x- = Math.sin(angle) * speed$
- The distance traveled by car on the $y - axis$ is given by $y- = Math.cos(angle) * speed$

B. Borders and Lanes of Road

This module looks at designing the road, the number of lanes for the road, different properties of the lane according to the car, and positioning of cars and vehicles on the roads.

1) Position of the road:

- The position of the car according to the car is determined and defined.
- The left border of the car is determined relative to the car's center of mass, and its value is the result of the subtraction of $x - coordinate$, and half of the width of the road; it is represented as below
- $left = x - width/2$
- The right border of the car is determined relative to the car's center of mass, and its value is the result of the addition of $x - coordinate$, and half of the width of the road; it is represented as below
- $right = x + width/2$
- The height of the car is a significantly large value and is extended in both the upward and downward directions.
- $constinfinity = 1000000$
- $top = -infinity$
- $bottom = infinity$
- The borders of the car are given by
- $borders = [[topLeft, bottomLeft], [topRight, bottomRight]$
- $topLeft = x : left, y : top$
- $topRight = x : right, y : top$
- $bottomLeft = x : left, y : bottom$
- $bottomRight = x : right, y : bottom$

2) Lanes:

- The lanes are equally divided on the road, and to determine the width of the lane, we can use the following formula, lane count is the number of lanes on the road $laneWidth = width/laneCount$
- To position all the lanes equally on the road, we can use the equation of a point which is t distance from starting of the line segment.
- $A + (B - A) * t$, where A represents the left border, B represents the right border, and t represents $thecurrentlane/totalnumberoflanes$

- t = $1/3$ the first lane on the road with 3 lanes. Then the first lane will end at $(2A + B/3)$ from the left border.
- Ideally, the car position should be at the center of the lane, and the center of the lane is calculated below
- $left + laneWidth/2 + Math.min(laneIndex, laneCount - 1) * laneWidth$

C. Sensors

Sensors are part of the vehicle, so this module takes all the car's properties and considers the following additional properties.

- rayCount - represents the number of sensors or the rays on the car.
- raySpread - represents how spread the sensors or the rays are; in other words, it's the angle between the first and last ray or sensor.
- rayAngle - represents the angle between the sensors or rays
- rayLength - represents the length of the sensor or the ray.
- The sensor or the rays is the line segment starting from the car's center of mass and extending to the length of the sensor, so the end points of the sensors can be calculated as below
- $x : car.x - Math.sin(rayAngle) * rayLength$
- $y : car.y - Math.cos(rayAngle) * rayLength$

D. Reading the borders

This module considers road borders, car borders, and traffic borders.

- The road borders can be obtained from the module Borders and Lanes and the borders are stored as
- $borders = [[topLeft, bottomLeft], [topRight, bottomRight]$
- Car Mechanics module has represented the car in the coordinate plane as the point and is the car's center of mass. This module determines the borders of the car and the traffic.
- The shape of the car or the vehicle in this project is considered to be a rectangle with height h and width w; using the Pythagorean theorem, the diagonal of the rectangle is $Math.hypot(width, height)$
- The distance between the car's center and its borders is constant, and its value can be calculated as $Math.hypot(width, height)/2$
- The angle between the diagonal and height is $alpha$ is given by $alpha = Math.atan2(width, height)$
- From the above-defined angle and diagonal, we can determine the borders below
- Top right vertex $x : x - Math.sin(angle - alpha) * rad$
- $y : y - Math.cos(angle - alpha) * rad$
- top left vertex $x : x - Math.sin(angle + alpha) * rad$
- $y : y - Math.cos(angle + alpha) * rad$
- Bottom right vertex $x : x - Math.sin(Math.PI + angle - alpha) * rad$
- $y : y - Math.cos(Math.PI + angle - alpha) * rad$
- bottom Left vertex $x : x - Math.sin(Math.PI + angle + alpha) * rad$
- $y : y - Math.cos(Math.PI + angle + alpha) * rad$

E. Collision Detection

This module considers the road borders, car borders, and traffic borders, and checks if there is any intersection between them. If there is any intersection, the sensors transfer this information to the car and leave the decision to the 'AI.' The following few points represent the underlying math to find the intersections.

- Consider the points $A(x1, y1)$ and point $B(x2, y2)$, any point on the line segment AB can be found as $P(x1 + (x2 - x1) * t, y1 + (y2 - y1) * t)$ where t being a real number. $Ift => [0, 1]$ then P will be between A and B.
- $Ift = 0.5, P((x1 + x2)/2, (y1 + y2)/2)$which is the mid point of AB.
- $Ift = 0, P(x1, y1) => A$
- $Ift = 1, P(x2, y2) => B$
- Consider another point $C(x3, y3)$ and point $D(x4, y4)$
- Any point $Q(x3 + (x4 - x3) * u, y3 + (y4 - y3) * u)$ on the line segment CD
- Let I be an intersection between the AB and CD.
- $Ix = Ax + (Bx - Ax) * t = Cx + (Dx–Cx) * u$
- $Iy = Ay + (By - Ay) * t = Cy + (Dy–Cy) * u$
- $(Ax - Cx) + (Bx - Ax) * t = (Dx - Cx) * u - -1$
- $(Ay - Cy) + (By - Ay) * t = (Dy - Cy) * u - -2$
- From 1 and 2, the value of t and u are
- $t = ((Dx - Cx)(Ay - Cy) - (Dy - Cy)(Ax - Cx))/((Dy - Cy)(Bx - Ax) - (Dx - Cx)(By - Ay)))$
- $u = ((Cy - Ay)(Ax - Bx) - (Cx - Ax)(Ay - By))/((Dy - Cy)(Bx - Ax) - (Dx - Cx)(By - Ay)))$
- The points I represents the intersection between the line segment AB and line segment CD and is at t units from point A and u units from point C
- The borders of the road, car, and traffic are the inputs for the readings, and the output will be the coordinates of the point of intersection and its distance(offset) to the point of the car's center. If there is no intersection close by, then the readings will be zero.
- These readings are the inputs of the Neural Network. The offset is subtracted from the one and sent to the Node. The underlying reason is that offset represents the distance from the point of intersection, and if the distance is close to zero, then there will be an imminent danger. If the distance is significant, then the nodes can relax.

F. Traffic

The Traffic module has almost the same properties as the car module; the key differences between the car and traffic modules are speed, control type, and color. The below points represent the properties of the traffic module.

1) Position of the car:

- The parameter defines the placement of the car in the coordinate plane. It is represented by the points x, y,

where x refers to the relative position on the x-axis and y represents the relative position on the y-axis.

- The position of the car refers to the point in the coordinate plane and will be at the center of the car's mass.

2) Width and Height:

- The Width represents the extension of the car on the x-axis.
- The car width is extended to the left and right of the car's center of mass.
- The Height represents the extension of the car on the y-axis.
- The car height is extended in the upward and downward directions of the car's center of mass.

3) Control type: This project considers three types of controls for the car.

- Traffic - If the default settings control the vehicle, movement with constant speed in the forward or the backward directions or if the car is resting on the road.

4) Speed:

- The speed refers to the traffic's maximum speed in the forward and reverse directions and is maintained at a fixed speed.

5) Color:

- If the control type of the vehicle is traffic, then the color will be randomized.

6) Controls:

- The movement of the car in forward and reverse directions.

G. Neural Network

Neural networks reflect the human brain, allowing computer programs to recognize patterns and solve problems in AI, machine learning, and deep learning. Neural networks contain node layers, an input layer, one or more hidden layers, and an output layer. Each node connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, it remains deactivated without sending any data.

1) Input Layer:

- Input nodes provide information from the outside world to the network. The Input nodes pass the information to the hidden nodes without performing any computation. In the current scenario, the input layer receives the data from the sensors. This data contains information about the intersection of either car and road borders and/or the car and traffic. If there are any intersections, the distance from the car's center to the intersection is recorded and later used.

2) Hidden Layer:

- The Hidden nodes have no direct connection with the outside world. They perform computations and transfer information from the input to the output node. A collection of hidden nodes forms a "Hidden Layer." While a feedforward network will only have a single input layer and a single output layer, it can have zero or multiple Hidden Layers. The values for the nodes in the hidden layer are the summation of the product of randomly generated weights in the range [-1, 1] and the input values. The node is activated if the summation of all such products is more significant than the bias.
- Values for the neurons in the hidden layer are products of a given input and its corresponding weight to the output and are calculated as
- $sum+ = level.inputs[j] * level.weights[j][i]$
- If the sum is greater than the bias of the neuron/output, then the neuron/output will be fired; else, the neuron/output remains deactivated.
- $if(sum > level.biases[i])level.outputs[i] = 1$

3) Output layer:

- The Output nodes are collectively responsible for computations and transferring information from the network to the outside world. The direction the car should move is determined by the outputs generated from the Network.
- $if(useBrain)$
- $controls.forward = outputs[0]$
- $controls.left = outputs[1]$
- $controls.right = outputs[2]$
- $controls.reverse = outputs[3]$

4) Bias and Weights:

- Randomly generated values between the range [-1, 1]
- $level.weights[i][j] = Math.random() * 2 - 1$
- $level.biases[i] = Math.random() * 2 - 1$

H. Network visualization

Visualizing the feedback from the output values is very interesting compared to understanding the values. This module will demonstrate visualizing the feedback from the neurons and visualize the neuron firing. The module follows two color patterns; yellow represents the positive values, and blue represents the negative values of the neuron. The dotted lines on top of the nodes in hidden, and the output layer represents the biases and follows the same color pattern. The input nodes are at the bottom of the window. These neurons get activated when there are readings from the sensors, and as the neuron gets closer to the intersection point, the nodes start firing up to yellow. The offset from the previous modules helps the Node to get brighter. The readings from the input layer are pushed to the hidden layer. The color patterns on the hidden layer are the summation of the product of randomly generated values of the weights and inputs from the previous layer. If these values are positive, the nodes appear yellow; if they are negative, they appear blue. The dashed lines with the yellow or blue color pattern represent the randomly generated bais on the hidden layer. The output is at the top with the directions imprinted on them. These directions are sent to the car to evade the traffic and road borders. If 'AI' gets close to the traffic right before it, then a neuron representing the reverse direction is fired, and 'AI.' slows down and changes its direction to either left or right depending on the values of the readings.

I. Genetic Algorithm

A genetic algorithm is a meta-heuristic inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection, where the fittest individuals are selected for reproduction to produce offspring of the next generation. Genetic algorithms are commonly used to generate high-quality optimization and search problem solutions. On the general scale, the Genetic Algorithm has the following phases.

1) Initialization:

- Randomly generated population in the given range, and the size generally depends on the problem. In some specific cases, the population is generated in the search space where it is highly likely to find the optimal solutions.
- In this project, the initial population will be all the readings from the sensors whose offset is close to one; when the offset values are close to one, it has an imminent danger of collision.

2) Fitness Function:

- Determines how fit an individual is and gives a fitness score; it's an ability of an individual to compete, and the probability that an individual will be selected for reproduction is based on its fitness score.
- In this project, a fitness function is required to evaluate the best car from the parallelization, and multiple optimal fitness functions can be created, such as the maximum distance covered by the car; the drawback of this approach is that the car can go backward without interacting with the traffic. The car can go in circles with a fixed center.
- The Minimum distance covered in the direction of the y-axis. While representing the simulation in the computer, the positive value of y is moving in the downward direction; hence, to achieve the maximum route without collisions is to get the car with a minimum value of y.
- The following math is used to calculate the fitness function. Find the minimum offset from all the offsets of the inputs.
  $bestCar = cars.find(c => c.fitness == Math.min(...cars.map(c => c.fitness)))$
- Once the best-fit car is found, the user can save it in the local storage and is used further in the mutation phase.

3) Selection:

- Selects the fittest individuals and lets them pass their genes to the next generation. If the generation is zero, the parents are randomly selected from the population. Two pairs of individuals (parents) are selected based on their fitness scores. Random Selection – the parents are selected randomly and not recommended, except for generation 0. Tournament Selection - Provides selection pressure by holding a tournament among S competitors in each group until it converges to 2 winners from all the groups, with S being the tournament size.

4) Crossover:

- Offsprings are created by exchanging the parents' genes among themselves until the crossover point is reached.

5) Mutation:

- Mutation occurs to maintain diversity within the population and prevent premature convergence. The mutation factor determines the percentage of mutation an offspring can undergo.
- In this project, the best car generated from the fitness function is saved for the mutation. The mutation factor is the percentage that represents the offspring's diversity, where one is 100% new, and zero represents a replica.
- Once the traffic is included in the simulation; if no best car is found in the local storage, the 'AI' starts with the mutation amount of 1 until the best car is found or can also try with different mutation factors to the randomly generated values of weights and bias. If the best car is saved in the local storage, the user can try different mutation factors to diversify the values generated from the weights and biases.

6) Merging the offsprings with the population:

- The offspring compete with the parents, and after every generation, the performance and diversity of the offspring are measured. The algorithm terminates if the diversity and performance of the offspring remain constant.

7) Termination:

- If there is no diversity or reached a certain generation limit.

8) Parallelization:

- The process of distributing the neural network across different processors. To achieve quicker and more reliable results, one of the best solutions is to make a copy of the neural network brain on each car and run the simulations.
- Given that the user has enough memory and GPUs on the local device, one can select the number of parallel brains used on the 'AI.' The project has provided a wide range of values to be used for breeding the best 'AI.'

IV. Results

- Fig.1 represents the car and results in the console in the direction of movement
- Fig.2 represents the borders of the road, the lanes on the road, and the car's position on the road at the center of the lane.
- Fig.3 represents the car and the sensors.
- Fig.4 represents the car and the sensors.
- Fig.5 represents the car and the sensors.
- Fig.6 represents the collision detection.
- Fig.7 represents the traffic.
- Fig.8 represents the traffic.
- Fig.9 represents the traffic.
- Fig.10 represents the network visulazation.
- Fig.11 represents the network visulazation.
- Fig.12 represents the optimization using the genetic algorithm.
- Fig.13 represents the optimization using the genetic algorithm.

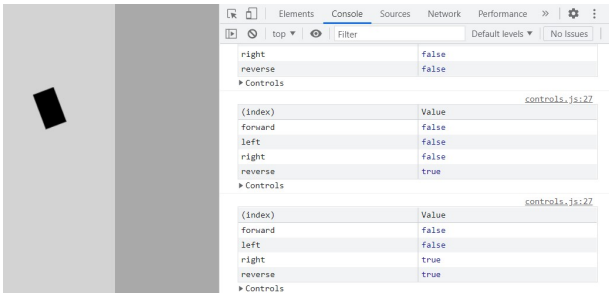

Fig. 1. The figure represents the car and results in the console in the direction of movement.

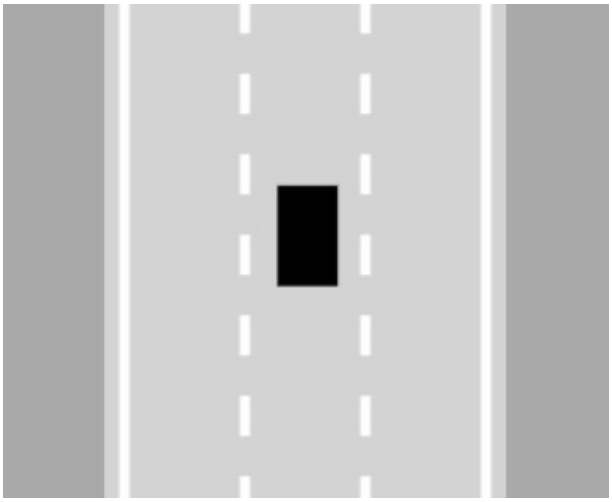

Fig. 2. The figure represents the borders and lanes.

- Fig.14 represents the optimization using the genetic algorithm.

V. Future Work

- Considering the weight of the car will way in many empirical challenges.
- Considering the lane speed, signals while changing lanes, automatic lights synchronized with the clock, oncoming traffic, merging lanes, and reading the signals on the road.
- This project just considered the straight-line road and paved the way for the curved, sharp turns, roundabouts, and many more real-life scenarios.
- Using other techniques of a neural network to initialize the values for weights and bias.
- Fitness function that rewards the car by maintaining the course in a single lane, following the speed regulations, and penalizes for unnecessary lane changes.
- Regarding other selection algorithms and drawing the comparison for the selection of the parents and offspring.
- Implementing the cross-over algorithm.

VI. Acknowledgment

I am especially indebted to Dr. Yanqing Zhang, a Georgia State University professor who has supported my career goals and worked actively to provide me with the protected academic time to pursue my goals. I am thankful to Dr. Ying Zhu, the Georgia State University professor and member of the Defence Committee, for accepting to be on the defense committee and for his swift actions and help in scheduling the meeting.

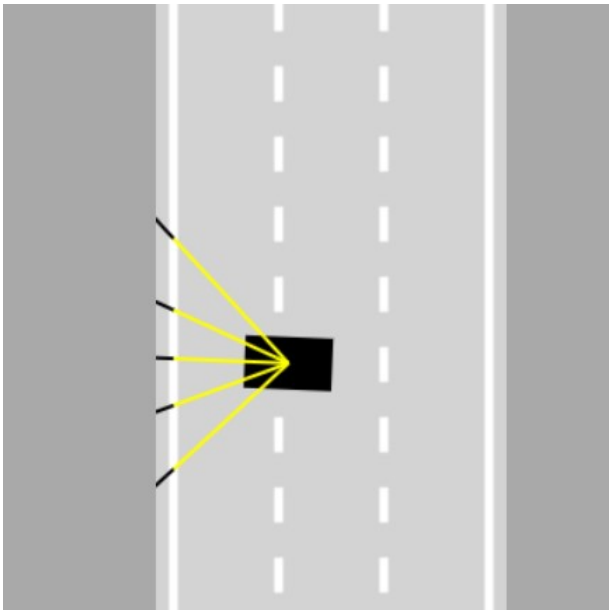

Fig. 3. The figure represents all the sensors activated.

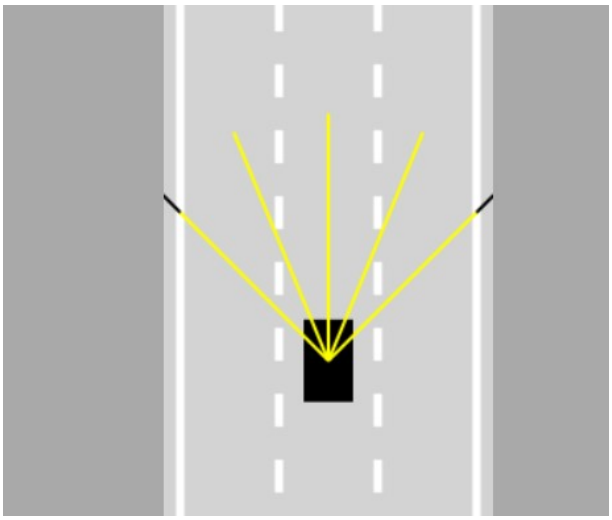

Fig. 4. The figure represents two of the sensors activated.

VII. Conclusion

In this project, we explored different modules involved in the self-driving car and their importance in bringing out the car with the intelligence to make its decision. Deep learning algorithms are one of the most desirable research topics in self-driving cars, including theory and application research. Deep learning will further enhance the intelligence and autonomy of self-driving cars and can solve some bottlenecks of traditional technologies, such as accuracy, robustness, and safety. This

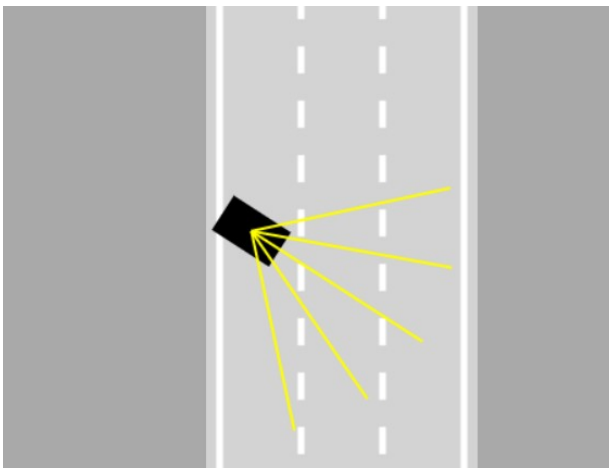

Fig. 5. The figure represents none of the sensors activated.

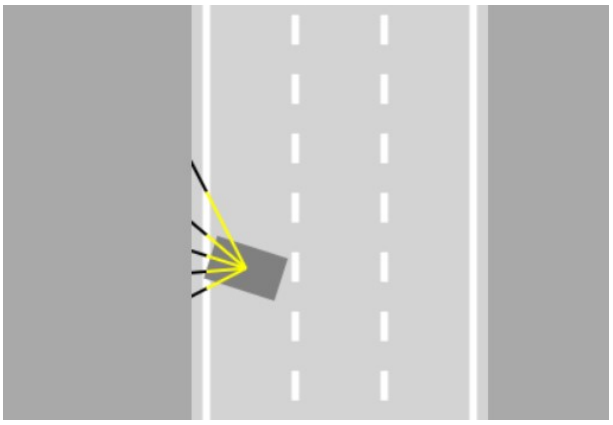

Fig. 6. The figure represents the collision detection by the car with the borders

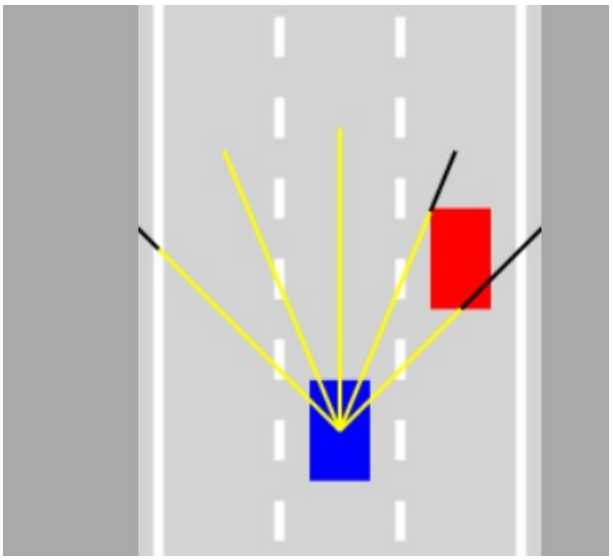

Fig. 7. The figure represents the traffic.

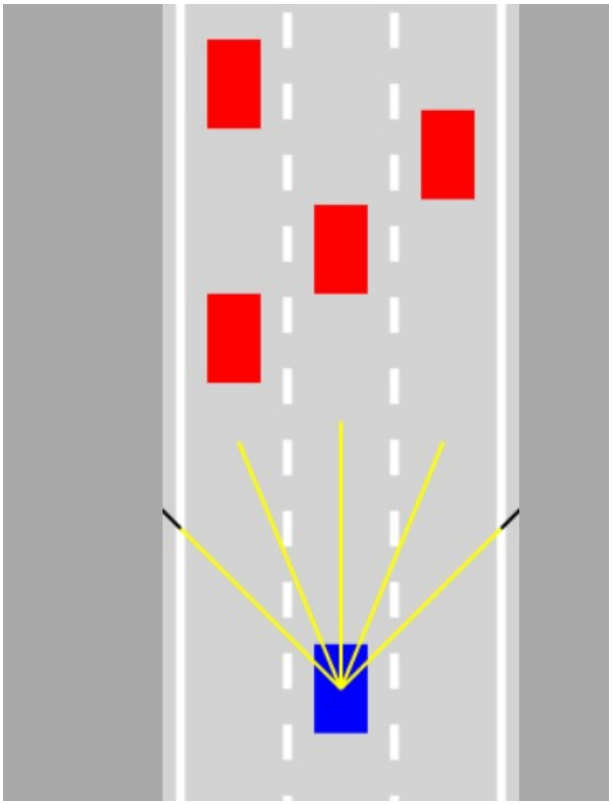

Fig. 8. The figure represents the traffic.

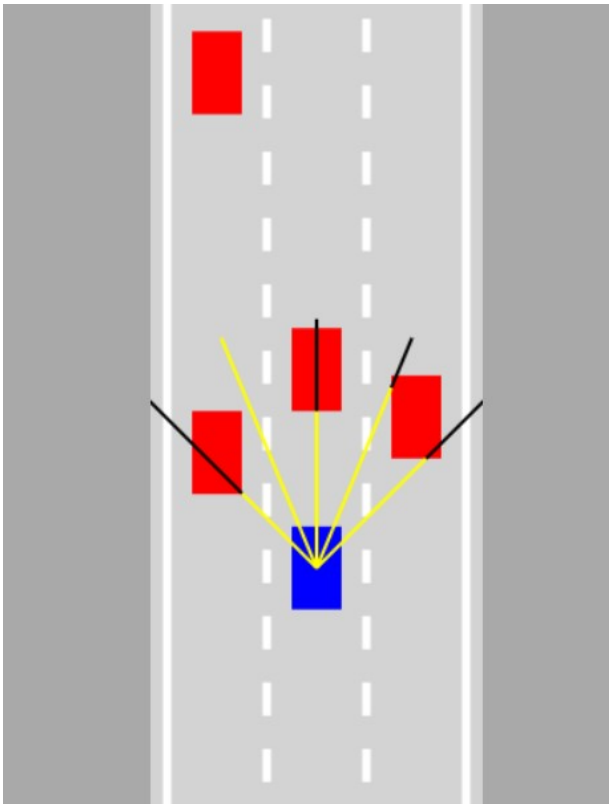

Fig. 9. The figure represents the traffic.

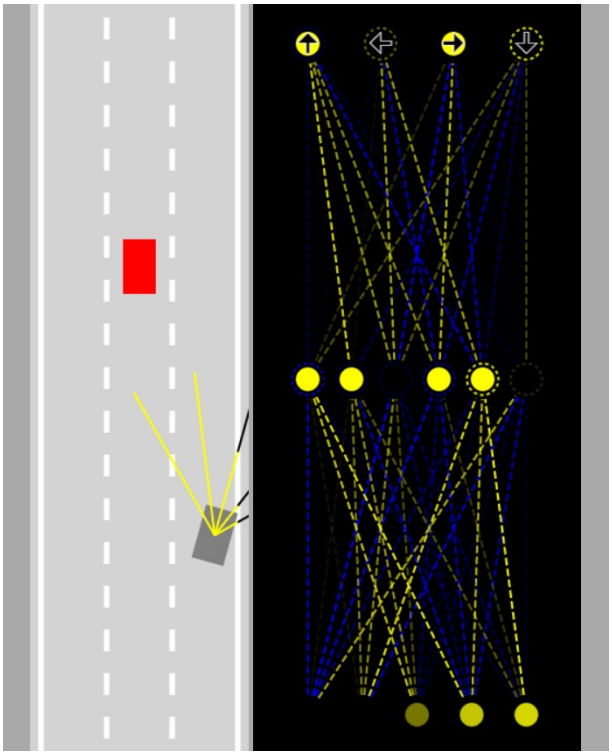

Fig. 10. The figure represents the damaged car, and the visualization represents movement in forward and right directions.

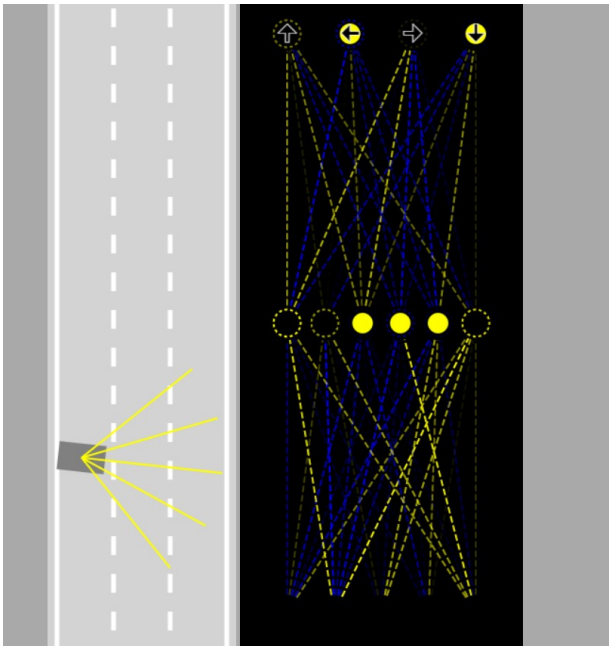

Fig. 11. The figure represents the damaged car, and the visualization represents movement in backward and reverse directions.

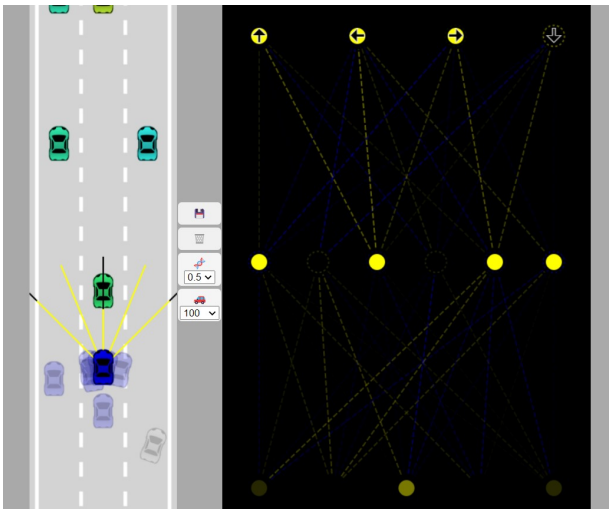

Fig. 12. The figure represents the concept of parallelization on 100 'AI cars' with a mutation rate of 0.5.

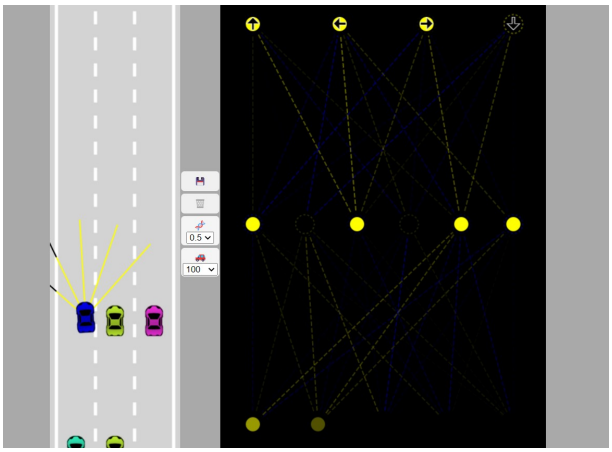

Fig. 13. The figure represents the concept of parallelization on 100 'AI cars' with a mutation rate of 0.5.

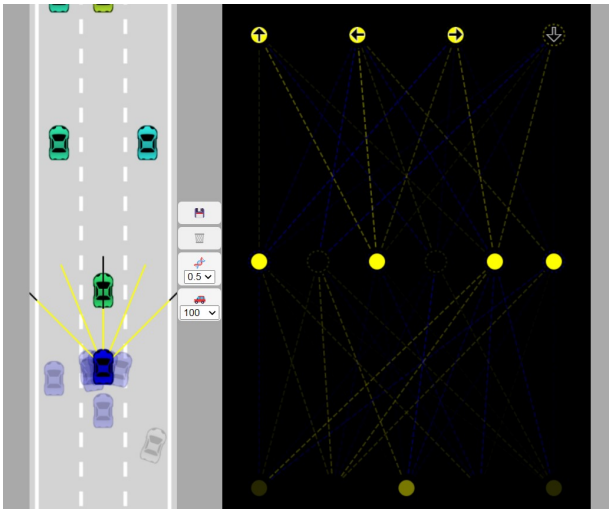

Fig. 14. The figure represents the concept of parallelization on 10 'AI cars' with a mutation rate of 0.5.

project explored the fundamental implementation of self-driving cars based on deep learning, and the results are exciting enough to demonstrate the potential of deep learning.

VIII. References

References

[1] Eureka. E! 45: Programme for a european traffic system with highest efficiency and unprecedented safety. https://www.eurekanetwork.org/project/id/45. [Retrieved May 19, 2019].

[2] B. Ulmer, "Vita ii-active collision avoidance in real traffic," in Intelligent Vehicles' 94 Symposium, Proceedings of the. IEEE, 1994, pp. 1–6.

[3] M. Buehler, K. Iagnemma, and S. Singh, The 2005 DARPA grand challenge: the great robot race. Springer, 2007.

[4] M. Buehler, K. Iagnemma, and S. Singh, The DARPA urban challenge: autonomous vehicles in city traffic. Springer, 2009.

[5] A. Broggi, P. Cerri, M. Felisa, M. C. Laghi, L. Mazzei, and P. P. Porta, "The vislab intercontinental autonomous challenge: an extensive test for a platoon of intelligent vehicles," International Journal of Vehicle Autonomous Systems, vol. 10, no. 3, pp. 147–164, 2012.

[6] A. Broggi, P. Cerri, S. Debattisti, M. C. Laghi, P. Medici, D. Molinari, M. Panciroli, and A. Prioletti, "Proud-public road urban driverless-car test," IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 6, pp. 3508–3519, 2015.

[7] P. Cerri, G. Soprani, P. Zani, J. Choi, J. Lee, D. Kim, K. Yi, and A. Broggi, "Computer vision at the hyundai autonomous challenge," in 14th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2011, pp. 777–783.

[8] C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. H. Bengtsson, and J. Didoff, "The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles," IEEE Wireless Communications, vol. 23, no. 4, pp. 146–152, 2016.

[9] R. McAllister, Y. Gal, A. Kendall, M. Van Der Wilk, A. Shah, R. Cipolla, and A. V. Weller, "Concrete problems for autonomous vehicle safety: advantages of bayesian deep learning." International Joint Conferences on Artificial Intelligence, Inc., 2017.

[10] Yaqoob, I.; Khan, L.U.; Kazmi, S.M.A.; Imran, M.; Guizani, N.; Hong, C.S. Autonomous Driving Cars in Smart Cities: Recent Advances, Requirements, and Challenges. IEEE Netw. 2019, 34, 174–181.

[11] Yurtsever, E.; Lambert, J.; Carballo, A.; Takeda, K. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. IEEE Access 2020, 8, 58443–58469.

[12] Gandia, R.M.; Antonialli, F.; Cavazza, B.H.; Neto, A.M.; de Lima, D.A.; Sugano, J.Y.; Nicolai, I.; Zambalde, A.L. Autonomous vehicles: scientometric and bibliometric review. Transp. Rev. 2018, 39, 9–28.

[13] Hussain, R.; Zeadally, S. Autonomous Cars: Research Results, Issues, and Future Challenges. IEEE Commun. Surv. Tutorials 2018, 21, 1275–1313

[14] 10. 10. Faisal, A.; Kamruzzaman, M.; Yigitcanlar, T.; Currie, G. Understanding autonomous vehicles: A systematic literature review on capability, impact, planning and policy. J. Transp. Land Use 2019, 12, 45–72. https://doi.org10.5198/jtlu.2019.1405.

[15] Greenblatt, N.A. Self-driving cars and the law. IEEE Spect. 2016, 53, 46–51.

[16] Birdsall, M. Google and ITE: The road ahead for self-driving cars. ITE J. (Inst. Transp. Eng.) 2014, 84, 36–39.

[17] Dikmen, M.; Burns, C. Trust in autonomous vehicles: The case of Tesla autopilot and summon. In Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017, Banff, AB, Canada, 5–8 October 2017; pp. 1093–1098.

[18] Coelingh, E.; Nilsson, J.; Buffum, J. Driving tests for self-driving cars. IEEE Spectr. 2018, 55, 41–45.

[19] Park, J.; Nam, C.; Kim, H.j. Exploring the key services and players in the smart car market. Telecommun. Policy 2019, 43, 101819.

[20] Toschi, A.; Sanic, M.; Leng, J.; Chen, Q.; Wang, C.; Guo, M. Characterizing perception module performance and robustness in production-scale autonomous driving system. In Proceedings of the 16th IFIP WG 10.3 International Conference on Network and Parallel Computing, Hohhot, China, 23–24 August 2019; pp. 235–247.