Introducing Autonomous Car Methodology in WSN

Promita Maitra[1], Sohini Nandi[2], Ipsita Saha[3], Poojarini Mitra[4], Sayani Chandra[5]

[1]Student, Dept. of Computer Science and Engineering, Gurunanak Institute of Technology, WBUT
[1]157/F, Nilgunj Road, Sodepur, Panihati, Kolkata - 700114, West Bengal, India
[2]Student, Dept. of Computer Science and Engineering, Gurunanak Institute of Technology, WBUT
[2]157/F, Nilgunj Road, Sodepur, Panihati, Kolkata - 700114, West Bengal, India
[3]Assistant Professor, Dept. of Computer Science and Engineering, Gurunanak Institute of Technology, WBUT
[3]157/F, Nilgunj Road, Sodepur, Panihati, Kolkata - 700114, West Bengal, India
[4]3Assistant Professor, Dept. of Computer Science and Engineering, Gurunanak Institute of Technology, WBUT
[4]157/F, Nilgunj Road, Sodepur, Panihati, Kolkata - 700114, West Bengal, India
[5]3Assistant Professor, Dept. of Computer Science and Engineering, Gurunanak Institute of Technology, WBUT
[5]157/F, Nilgunj Road, Sodepur, Panihati, Kolkata - 700114, West Bengal, India

Abstract - An autonomous car, also known as a driverless car, self-driving car or robot car, is an autonomous vehicle capable of fulfilling the human transportation capabilities of a traditional car. As an autonomous vehicle, it is capable of sensing its environment and navigating without human input. Robotic cars exist mainly as prototypes and demonstration systems, but are likely to become more widespread in the near future. Expert members of the Institute of Electrical and Electronics Engineers (IEEE) have estimated that up to 75% of all vehicles will be autonomous by 2040.

Collision avoidance is the great challenge in the active safety of the road vehicles. In India the total annual deaths due to road accidents has crossed 1.18 lakh, according to the latest report of National Crime Records Bureau (NCRB). In this report we describe part of the work conducted related to the studies of route finding and vehicle collision detection and removal algorithms in a situation where multiple vehicles are moving simultaneously.

This is a description of the functional capabilities of a series of computer programs that can be used to investigate the consequences of autonomous car methodology in WSN. Main three elements of the simulation tools for autonomous driving are 1. Shortest Route Finding, 2. Collision Detection, 3. Collision Avoidance

In this work it was deemed desirable to have an interface between the simulation programs so that users can take a note of the vehicle movement profiles while running in a multi-car scenario given the input cases for vehicle operations. Program functions and the interfaces between programs are outlined. Some simulation run cases are shown for various functions implementing user-specified map containing blocks. The screenshots captured during the case studies are also illustrated in this report.

Keywords - WSN, Autonomous Car, Route Finding, Collision Detection

I. INTRODUCTION

Wireless sensor networks (WSNs) have a number of applications such as environmental monitoring, target tracking, disaster recovery, and surveillance. A WSN typically consist of one or more sink nodes and a large number of sensor/relay nodes. As the research on WSNs continues to evolve, low-end Cameras are being adapted to extend the capability of sensing.

It becomes more challenging to control congestion, collision and find the shortest route in WSNs as more advanced sensors, such as CMOS cameras and microphones, are added to WSNs [1][2]. Thus, in this paper, we focus on finding the shortest possible route for a particular set of input and detect collisions while moving in a multi-car scenario possibly avoiding all the detected collisions based on the estimation in a WSN.

The contribution of our work is threefold: First, We develop new algorithms to estimate the hop-count for reaching destination from source using a particular algorithm and find the shortest from them. We consider all the static blocks mentioned in map while making this decision. Second, we opt for a multi-car scenario where more than one car can run simultaneously detecting possible collision in a one-hop radius. Third, local node control to modify the already decided route for possible collision avoidance technique if a collision is detected somewhere while running.

A. Advantages of Autonomous Car

- Fewer traffic collisions
- Increased roadway capacity and reduced traffic congestion (due to reduced need for safety gaps), and the ability to better manage traffic flow.

- Relief of vehicle occupants from driving and navigation chores.
- Higher speed limit for autonomous cars.
- Removal of constraints on occupants' state
- Alleviation of parking scarcity
- Elimination of redundant
- Reduction of space required for vehicle parking.
- Reduction in the need for traffic police and vehicle insurance.
- Reduction of physical road signage – autonomous cars could receive necessary communication electronically
- Smoother ride.

B. Wireless Sensor Network (WSN)

A wireless sensor network (WSN) consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling control of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.

C. Vehicular Ad-hoc Network (VANET)

A vehicular ad hoc network (VANET) uses cars as mobile nodes in a MANET to create a mobile network. A VANET turns every participating car into a wireless router or node, allowing cars approximately 100 to 300 metres of each other to connect and, in turn, create a network with a wide range. As cars fall out of the signal range and drop out of the network, other cars can join in, connecting vehicles to one another so that a mobile Internet is created.

It is estimated that the first systems that will integrate this technology are police and fire vehicles to communicate with each other for safety purposes. Intelligent vehicular ad-hoc network (InVANET) is another term for promoting vehicular networking. InVANET integrates multiple networking technologies such as Wi-Fi IEEE 802.11p, WAVE IEEE 1609, WiMAX IEEE 802.16, Bluetooth, IRA and ZigBee. Vehicular ad hoc networks are expected to implement wireless technologies such as dedicated short-range communications (DSRC) which is a type of Wi-Fi. Other candidate wireless technologies are cellular, satellite, and WiMAX. Vehicular ad hoc networks can be viewed as component of the intelligent transportation systems (ITS). As promoted in ITS, vehicles communicate with each other via inter-vehicle communication (IVC) as well as with roadside base stations via roadside-to-vehicle communication (RVC).

II. PROPOSED SIMULATION MODEL

A. Explanation of The Model

It was decided to represent the area as a n X n matrix format. This matrix has blocks and nodes. The blocks represent buildings, bridges and nodes represents cars or vehicles. Matrix cell position of blocks and the type of blocks i.e. buildings or bridges and start and end matrix cell position of that bridge are stored in a global database. When a car finds a block at its next cell position [3], it checks whether it is a building or bridge. If it's a building, then the car changes its direction or path; and if it's a bridge then it goes using it. The route using which the car is travelling is stored in another database, called local database. Any change in the route is updated on it.

First, the project started with just one node and some blocks. At this stage the start and destination cell positions were given to the code. Here three algorithms were used to reach the destination: in the first one, we first consider the row and move the car node along the node. If there is block at that row or the row number of destination matrix cell position is achieved, then we move the care along the column. In the second one, we first consider the column and then row; and in the third one we move the car node diagonally first, then if a building prevents its motion or destination row or column achieved then the row or column, which one is smaller in number is taken into consideration when the difference between current matrix cell position and destination matrix cell position row and column is found out. Using these three algorithms, the available paths are found and then the shortest one among them is picked.

On the second phase of software implementation two nodes and take run time input were taken. With the concept of Java threads, two threads implements two cars. In this stage users are given the permission or facility to add or remove or reposition blocks, i.e. buildings or bridges to the matrix.

On the third phase the collision detection system was developed. Here a car checks for another car or buildings within one block radius of it. If it is a building then it detects collision. If it's a car then it searches for the route of the other car in the local database. If the next matrix cell for both cars is common then it detects a possible collision. Till now only two nodes were considered, but from this phase users will be able to check with more than two cars with all of them having different sources and destinations.

On the fourth phase the collision avoidance system was developed. When a car detects a collision, it either automatically stops, allows the other car to pass by or it finds the current position of the other car having same next position. Then it chooses a position as its next which is not adjacent to the other car's current position.

B. The Main Model Algorithm

```
Start
Enter Total Number of Cars
Enter Source and Destination Position for each Car
Calculate routes following 3 algorithms
Find the shortest among them considering hop count
While (the Car is not at Destination)
        Calculate Next Position following the
        shortest Route
        Check if there is any collision possibility
        If (Collision)
                Find the second Car's current
        position
                Remove Collision by changing first
                Car's Next Position
        End If
        Update Next Position in Database
        Move the Car in the calculated Next Position
End While
End
```

C. Pseudo code for Route-Finding Algorithms

1) Row-First Algorithm:

```
Row:
While (Current Row != Destination Row)
        Try to proceed along the row first to reach
destination row
        If(Block)
                Goto Column
        End if
End While
Column:
While (Current Column != Destination Column)
        Try to proceed along the row first to reach
destination column
        If(Block)
                Goto Row
        End if
End While
```

2) Column-First Algorithm:

```
Column:
While (Current Column != Destination Column)
        Try to proceed along the row first to reach
destination column
        If(Block)
                Goto Row
        End if
End While
Row:
While (Current Row != Destination Row)
        Try to proceed along the row first to reach
destination row
        If(Block)
                Goto Column
        End if
End While
```

3) Diagonal-First Algorithm:

```
While (Current Column and Row != Destination
Column and Row)
        Try to proceed to the adjacent diagonal cell
        If(Block)
                If (Current Column != Destination
                Column)
                        Follow Column-First
                        Algorithm to find next
                Else
                        Follow Row-First
                        Algorithm to find next
                End If
        End if
End While
```

4) Pseudo code for Collision-Detection Algorithm:

```
Get Next Position of the Car
Check in Global Database for a match in Next
Position
If (Match Found)
        Generate Collision Alert
        Modify Next Position with Collision
        Removal Algorithm
Else
        Continue moving the car in the Next Position
End If
```

5) Pseudo code for Collision-Removal Algorithm

```
Get Current Position of the Other Cars
For all the adjacent cells of First Car
        If (The cell is adjacent to Any Other Car)
                Avoid the Cell
        Else
                Choose the Cell as Next Position of
                First Car
                Update in database
                Break For Loop
        End If
End For
```

III. RESULTS

A. Comparison of three Route Finding Algorithms

To compare the time taken and the distance covered by the car using three different algorithms we developed, we give the same source and destination as input in a 10 x 10 matrix and plot a graph using the matrix cell position the car traversed. We took the Y axis of the graph as the route column and the X axis as the route row and plotted the graph. This graph clearly shows which route finding algorithm takes least time and traverses least amount of path to reach destination.

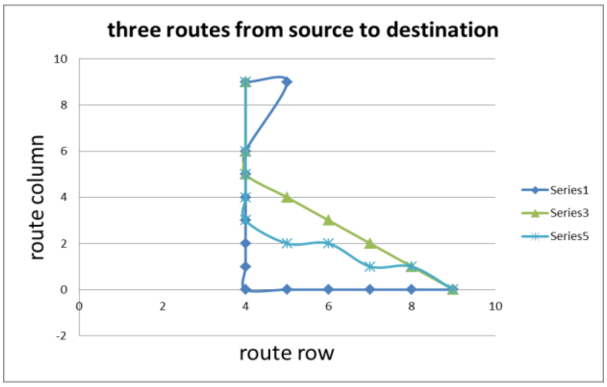

Fig. 1 Three routes from source to destination

B. Working of Collision Detection and Removal Algorithm

We have successfully implemented the dynamic scenario where the nodes can detect not only the static objects but also the moving cars as blocks and can adjust their next hop while driving in a particular chosen route.

The screenshots given below shows the working of the route finding algorithms considering all the static hops mentioned in the map and also outputs the hops that the cars have taken while running in multi-car scenario successfully detecting and avoiding collision. One of the screenshots also shows how the user can modify the map or make their own using the simulator.

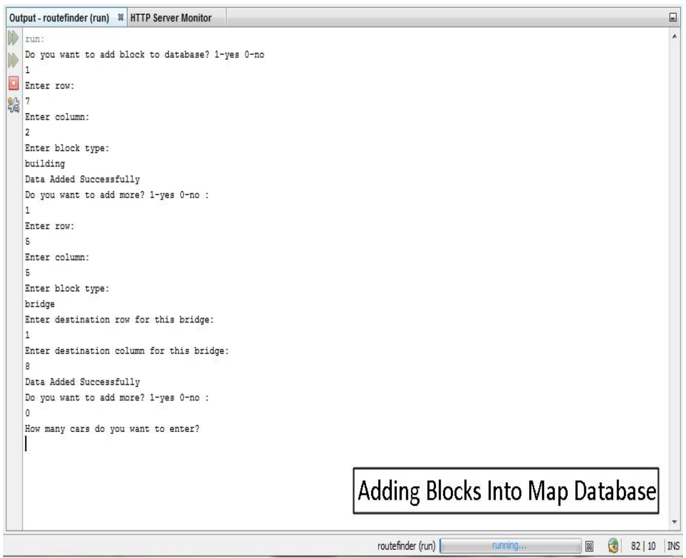

Fig. 2 Adding Blocks Into Map Database

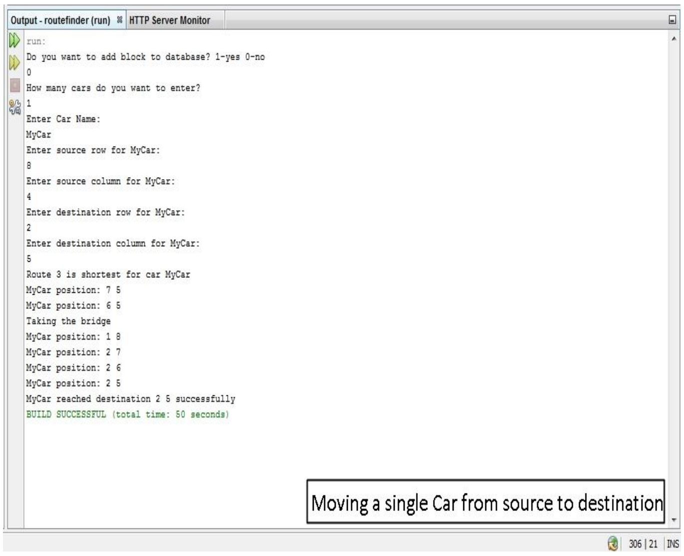

Fig. 3 Moving a single Car from source to destination

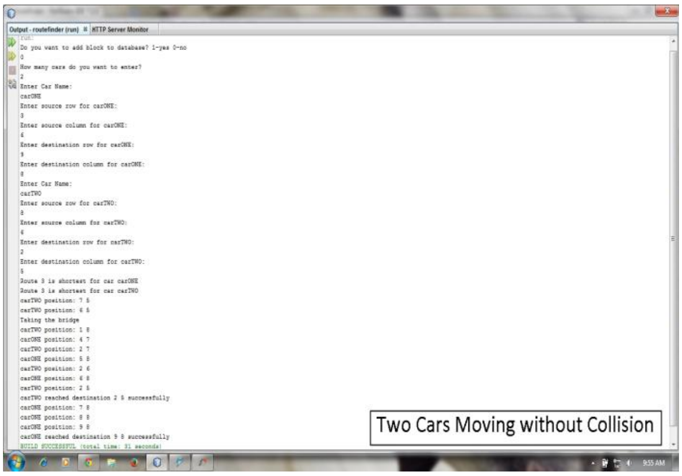

Fig. 4 Two Cars Mving without Collision

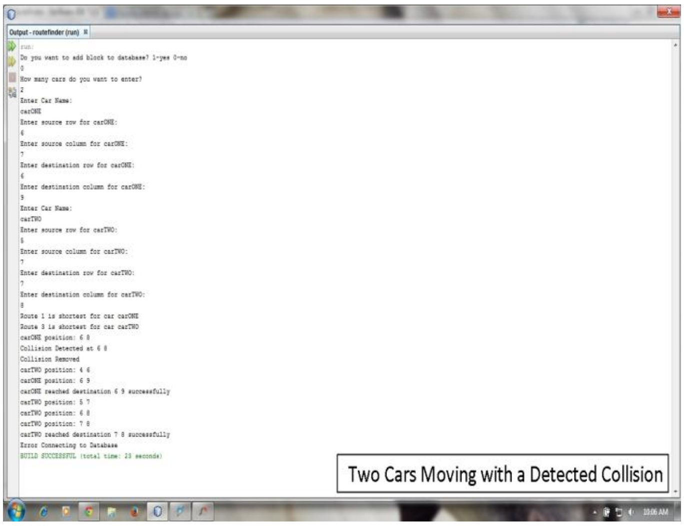

Fig. 5 Two Cars Moving with a Detected Collision

IV. CONCLUSION

In this software simulation, we mainly focused on three parts.

- Adding different types of blocks into database
- Finding shortest route from source to destination
- Detect and remove collision on way to destination

After finding three routes for a particular source-destination combination, we even did a comparative study among those algorithms plotting a graph to determine which the best one is. But we cannot omit a particular one because any of these three can be useful for a particular set of source and destination.

More features can be added to the existing algorithms, like a collision detection algorithm for curved path or a congestion detection technique after finding the shortest route.

REFERENCES

[1] J. Jeong, S. Guo, T. He, D.H.C. Du "Autonomous Passive Localization Algorithm for Road Sensor Networks", Proc. 11th World Congr. Intell. Transp. Syst., 2004

[2] A. Barth and U. Franke "Where will the oncoming vehicle be the next second?", Proc. IEEE Intell. Veh. Symp., pp.1068 - 1073 2008

[3] A. E. Broadhurst, S. Baker and T. Kanade "A prediction and planning framework for road safety analysis, obstacle avoidance and driver information", Proc. 11th World Congr. Intell. Transp. Syst., 2004