Autonomous Indoor Navigation Robot

Jayaparvathy R[#1], Sheeba Angel A[#2], Nitin Barattwaj P[#3], Srihaarika Vijjappu[#4], Srisruthi sridhar[#5]

[#] Department of Electronics and Communication Engineering,
SSN College of Engineering, Chennai, Tamil Nadu, India.
[1]jayaparvathyr@ssn.edu.in, [2]sheebaangel05@gmail.com, [3]nitubattywaj@gmail.com
[4]srihaarikavijjappu@gmail.com, [5]srisruthisridhar@gmail.com

***Abstract*—Autonomous robots are intelligent machines capable of performing tasks by themselves without human control. In this work, an autonomous robot that can navigate an indoor restaurant setting and assume the role of an automated customer servicing system is proposed. The robot has to be aware of its own position in the workspace and compute the shortest path to the customer when introduced in a known environment. The RRT path planning algorithm is optimized using goal biasing and vector field RRT is built upon it to reduce the total number of computations required, eventually reducing the latency. We closely model a real time restaurant environment where multiple customers request for service simultaneously or at random instants of time. The servicing robot has to prioritize requests based on different parameters such as distance-to-goal, number of obstacles in the path, frequency of calls made by the customer and other special concession for customers based on their age etc. and appropriately service the table with highest priority. The performance analysis of the proposed algorithm is compared with RRT based routing and it is observed that there is significant reduction in servicing time when the proposed model is implemented.**

Keyword – Path Planning, Localization, Navigation, Autonomous, RRT.

I. INTRODUCTION

The future proffers a world in which robots perform almost all kinds of jobs extending across the spectrum from dull, dirty, repetitive ones to highly specialized, intelligent decision making tasks. The role of humans is quickly getting displaced by robots across many industries and the prime reason for this rapid shift is due to the evolution of autonomy in robots aided by artificial intelligence. Autonomy is the ability to be self-aware, self-reliant and navigate the workspace at ease without human intervention. We have attempted to build an autonomous robot that can navigate an indoor restaurant setting and assume the role of an automated customer servicing system.

Systems that are aware and interact with their environment are autonomous systems. Although autonomous systems are prevalent, we try to envision autonomous systems of the future and explore the multitude of possibilities which could greatly enhance the quality of life while addressing the needs of people. The progressive technologies in machines, equipment's, sensors, computation and communication technologies, architecture, algorithms, security/trust lead us to design and develop fairly complex systems with relative ease to replace humans with machines, thereby improving performance, user satisfaction, throughput and eliminate flaws caused by humans. With the huge buzz around self-driving cars and drones for delivery, the smaller and less complex applications such as a waiter robot could be easily addressed and is taken up as a challenge. This could be the basis for more complex solutions with extremely promising outcome such as unmanned aerial vehicles (UAV's) or underwater robots which operate in a 3D environment.

Localization and autonomous navigation in an indoor environment have been in existing literature. Dong Jin Seo, et al.[2] explains a system that implements a localization module using the Monte Carlo localization algorithm based on random numbers to determine the effect of moving objects on the sensor data. For simulation purposes, particles are moved by a velocity motion model. The estimated value is derived from sensor model and the laser range sensor data.

Johann Borenstein, et al.[5]provide detailed information on odometry and explains how it provides easily accessible real-time positioning information for a mobile robot. The disadvantage is its unbounded accumulation of errors. The most dominant errors are 1) uncertainty about the effective wheelbase and 2)unequal wheel diameters. The authors propose a calibration technique that will increase the robot's odometric accuracy. Karaman, et al. [6]have discussed the various sampling based path planning algorithms - such as PRM , RRT, RRT*, RRG, the authors observed that the probability with which the RRT converges to an optimal solution, as the number of samples approaches infinity, is zero. RRT∗ maintains a tree structure and serves as the best choice among path planning algorithms.

According to Liang Ma, et al.in [7] a fixed sampling length is used to find new nodes in RRT algorithm. So even in an environment with very few obstacles it takes a minimum amount of computations before reaching the goal. To solve these problems, we go in for an aggressive extension strategy by means of which new sampling points are taken only in region towards goal point. So less time is wasted on scanning the entire environment. In case of any collisions, the trajectory till that point is saved. Sunhong Park, et al. [9] proposed an odometry based localization algorithm that leads to accumulation of error over time. The authors adopted a pattern of arranging the RFID tags on the floor to reduce the estimation error. The RFID system reads IC tags on the floor which allows the robot to roughly deduce the current location and pose of the robot.

The paper is organized as follows: Section II describes the path planning and the existing RRT algorithm. Section III explains optimization of the RRT* algorithm. Section IV provides the scheduling and queuing approaches. In section V, provides the algorithm for navigation and localization. Section VI, describes the hardware implementation of the autonomous robot. In Section VII, we provide the results and discussion. Section VIII provides the conclusion and scope for further extension of this work.

II. PATH PLANNING

A. Path Planning

Navigation and path planning play an important role to execute any task. Path planning is the main concept to design a mobile robot. Mostly, the complication of path planning is finding the paths by connecting various places in an environment as graph, maze and road. Path planning lead mobile robots to see the obstacle and generate an optimal solution so as to avoid them. These approaches help to produce a gradual improvement towards better performance in term of time, distance, cost and complexity.

Mobile robot path planning has a few main identities according to type of the environment, algorithm and completeness. The identities are based on whether it is fixed or changing, small or large and complete or analytic. The fixed path planning refers to the surroundings which consists of only fixed objects or obstacles other than a navigating robot and dynamic path planning refers to surroundings consists of movable objects. Meanwhile the small and large path planning works on algorithm where the data about the surrounding is available. In large Path planning, the details about the surroundings already feed through a database consists of map, cells, grid or etc. and in case of a small path planning, the robot has no details about the navigating surroundings and robot has to sense the surroundings before decides to move for obstacle avoidance and generate route planning toward destination. Depending upon the origin of track various types of obstacle avoidance algorithms are used.

B. RRT Algorithm

We propose a service robot for an Indoor environment which falls under the category of global path planning. The map of the room is already known to the robot and it makes use of tree based Motion Planner known as RRT Algorithm. A Rapidly-exploring Random Tree (RRT) is a path planning algorithm that is designed to quickly search large regions consisting of obstacles and support non-holonomic dynamics. RRTs are constructed incrementally by expanding the tree towards the unsearched areas by taking a randomly-sampled point in the configuration space while satisfying given constraints. As each sample point is drawn, a connection is attempted between it and the near state in the tree. The length of the connection between the tree and the new state is controlled by a growth factor known as sampling length. The position of the random samples controls the direction of the tree growth and the growth factor determines its rate. The key idea is to bias the search towards unexplored portions of the configuration space. The working of RRT algorithm can be divided into 2 phases

1) Expansion of tree over the workspace.

2) Finding out the shortest path to reach the goal.

- Selects sample points in space and checks if the randomly sampled point coincides with any obstacle.
- If the point does not fall inside any obstacle, the point is added to a tree which is made to grow throughout the workspace.

C. Advantages of RRT Algorithm

- A tree rapidly explores the whole area, instead of ‘staying’ in the neighborhood of the start node.
- The RRT algorithm is quite simple to program; so this quality leads to a fast analysis to find a path.
- It is faster than conventional graph based algorithms like Dijkstra's, A*, and so can be used for path planning purposes over large spaces of area.

D. Algorithm

Phase 1 - Expansion of the Tree:

- Initialize the tree by choosing the initial position of the robot as the root element as shown in fig.1.
- Predefine the sampling length and the total number of points to be sampled (Number of nodes in the tree).
- Now a random point is chosen from the configuration space and checked if the randomly sampled point coincides with any obstacles.
- Next the algorithm scans the entire tree to find the closest neighbor to the sampled random point.
- If the sampled point falls within sampling radius of the nearest neighbor, it is added into the tree.
- If the constraint is not satisfied, then a point at the distance equal to sample length in the direction of the new state is chosen and added to the tree. Thus results in expansion of the tree in that direction.
- It is also made sure that the line joining the newly sampled point and nearest neighbor in the tree does not collide with any obstacle.
- Then the above steps are iterated until the number of samples taken reaches the total number of points specified at the beginning.

2)Phase 2 - Determining the Final Path:

- Now the algorithm again starts from the root node and scans the entire tree to locate the goal point.
- If the goal point is not present, then the point closest to the goal location is taken.
- Once the Goal point is reached starting from the Goal the algorithm finds out the parent node and moves to it until the root node is reached.
- Along the process all the parent nodes are stored in an array and reversed, which constitutes the route the robot must move to reach the Destination. Thus RRT Algorithm gives the final path to be taken to reach the destination by avoiding all the obstacles in the given space.

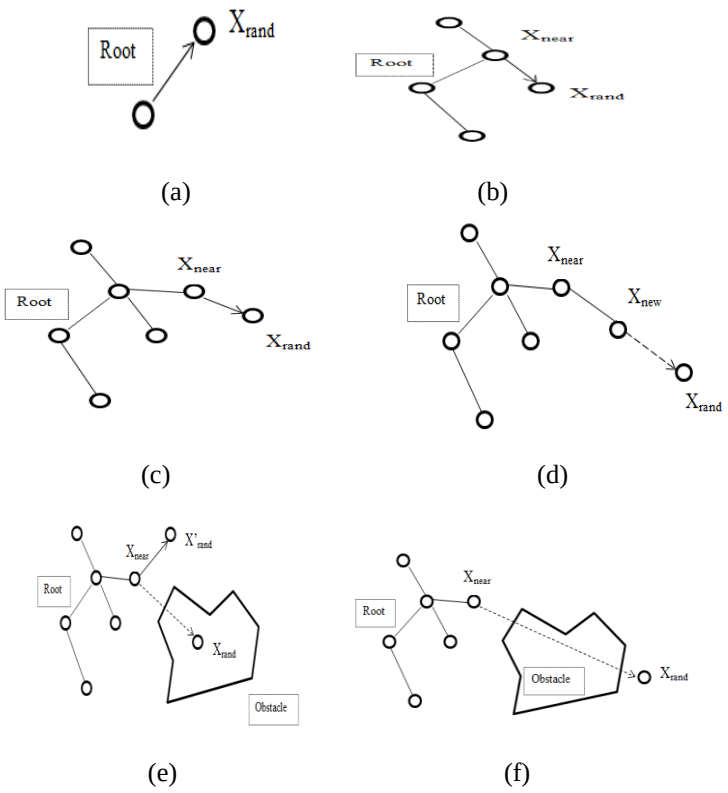

Fig. 1. Tree Growth (a) First x_{rand}(b) Expanding tree (c)x_{rand} within sample length (d)x_{rand} beyond sample length (e) Random point inside obstacle(f) line joining x_{rand} and x_{near} passes through obstacle.

E. Drawbacks of the RRT Algorithm

Although Rapidly-exploring Random Tree algorithm gives us a clear approach to reach the destination, it suffers from a lot of drawbacks.

- The algorithm is not deterministic. A found path is not the same the next time a path has to be found on the same start node.
- The given path to the Destination is not the best feasible one.
- Since the new states are sampled at random, there is no guarantee, it will reach the goal at all times when there are a lot of obstacles.
- The found path is sharp-edged, which cannot be driven easily.
- It also takes larger number of iterations to find the path, which consumes efficiency. Hence it is not very reliable for real time systems.

RRT alone may not be appropriate to solve a path planning problem for a mobile robot as it cannot incorporate additional cost information such as smoothness or length of the path.

III. RRT* WITH VARIANTS

A. Modifications in Existing RRT Algorithm

Motion planning algorithms, such as the actively searching Random Trees (RRTs), have been shown to perform efficiently in real time and to have logical ideas such as probabilistic completeness. However, logical bounds are not on the quality of the results obtained by these algorithms, example, in terms of a given cost function, have been established. It is shown that, under technical conditions, the cost of the optimal path produced by RRT meets to a worst values, as the number of trials were raised.

In order to address this issue, a new algorithm called the Rapidly-exploring Random Graph (RRG) was developed and it has been proven that the cost of the best path returned by RRG converges to the optimum almost surely. But the drawback with this approach is that the tree structure is lost and replaced with graph theory. Hence there is need to maintain the tree structure while preserving the asymptotic optimality of RRG. This is achieved by RRT*, an efficient incremental sampling based algorithm that maintains the tree structure and is easy to implement along with provable optimality properties.

*B. RRT**

Sampling-based motion planning approaches such as PRM Probabilistic road map, RRT, EST etc. have been widely used in motion planning problems involving complex kinematic and differential constraints. An iteration of one such algorithm is the RRT*. Like the RRT, it quickly finds a feasible motion plan. Further, it enhances the plan toward the best solution in completing the plan execution within a remaining time. This refinement property is advantageous, as most robotic systems take significantly more time to execute trajectories than to plan them. For instance, robotic cars spends only a few seconds in path planning and takes some minutes to drive towards the goal. In that surroundings, asymptotic optimality is particularly helpful, the remaining part of the planned path is enhanced by moving along in its route in available computation time.

- The RRT∗ starts with an empty tree and adds a single node corresponding to the initial state.
- It then builds and refines the tree through a set of N iterations. The RRT* cumulatively constructs the tree similar to the RRT by considering the random sampling state from the obstacle free space and solving for a route X_{new} that prolong the nearest node in the tree $z_{nearest}$ into the sample.
- If this trajectory does not collide with obstacles, the standard RRT inserts the new node z_{new} into the tree with $z_{nearest}$ its parent and continues with the next iteration.
- The operation of RRT* changes in choosing the closest node as the parent node. It takes all the nodes surrounding the z_{new} and calculates the cost of each node for choosing a parent node.
- This process evaluates the total cost as the additive combination of the cost associated with reaching the potential parent node and the cost of the trajectory to z_{new}.

Fig. 2. Path obtained from RRT

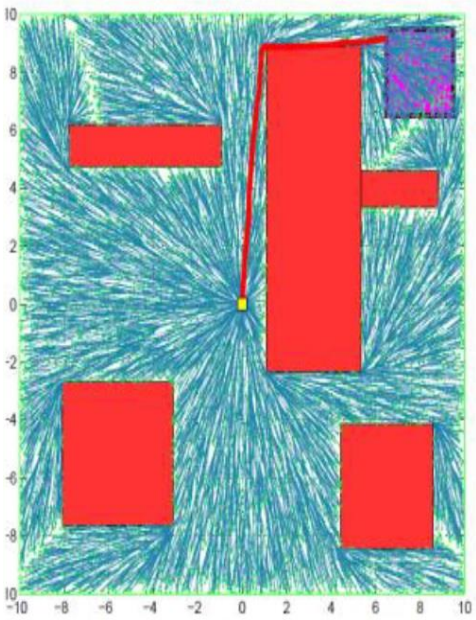

Fig. 3. Path obtained from RRT*

- The node that yields the lowest cost becomes the parent as the new node is added to the tree. The Rewire procedure then checks each node z_{near} in the vicinity of z_{new} to see whether reaching z_{near} via z_{new} would achieve lower cost than doing so view its current parent.
- The total cost confederating with z_{new} changes/ rewires by reducing the connections. The RRT$*$ then proceeds with the consequent number of iteration.

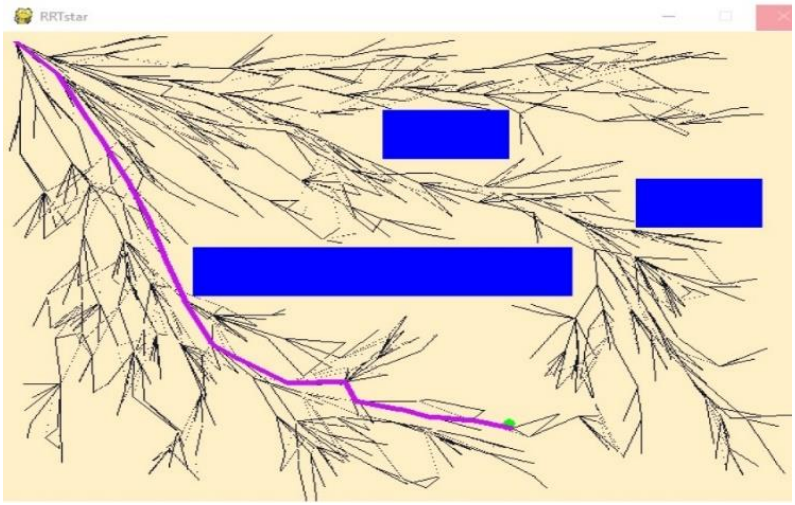

Fig. 4. The growth of tree using the original RRT algorithm without optimizations, number of computations =865

*C. Optimizations of RRT**

Although the RRT* algorithm returns an optimal solution, it takes a large number of computations before the algorithm finds the goal. This affects the speed of the program as well. Hence to make the RRT* algorithm time optimized and reduce the latency, certain improvements have been made on the existing framework of the algorithm.

1) Goal Biasing: To increase the probability of hitting the goal point, variants of RRT like goal bias or bidirectional tree growth is used. On selecting the random point, the goal bias modification simply selects a point near the goal with pre-fixed frequency, e.g. once every five iterations .A region around the goal is defined and during the generation of random samples to expand the tree, a point from within the goal region is arbitrarily chosen every few, tens or hundreds of iterations. With this simple modification, we can make the expansion of the tree biased to the goal. The random point chosen from the goal region might save few additional computations which the algorithm would otherwise have taken if the goal region were not considered and used as sample space.

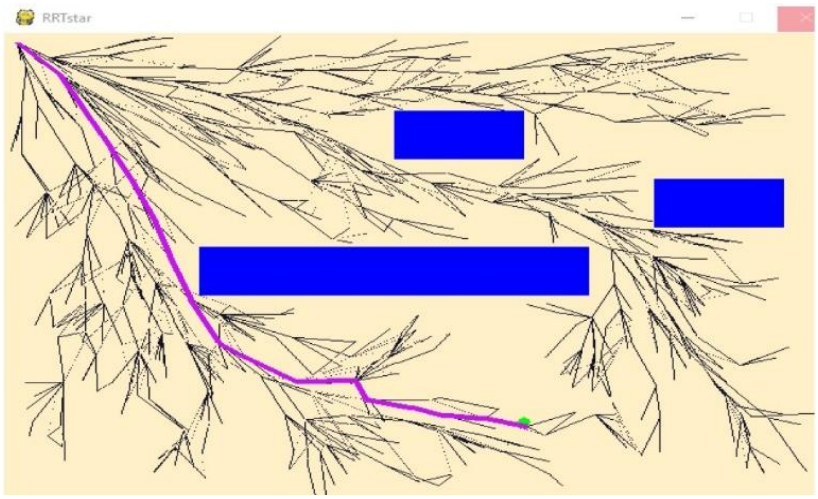

Fig. 5. The growth of tree using optimized RRT* with goal bias and the number of computations has reduced to 266.

2) Aggressive Growth: In aggressive growth strategy the orientation of the goal with respect to the start location is found by computing the slope between the two points. The growth of the tree is restricted to the following angle range:

slope + offset angle

slope– offset angle

The offset is typically few degrees greater than/lesser than the slope angle. This process reduces the unwanted expansion of tree in directions other than the direction of the goal point, thus minimizing the latency in tree growth.

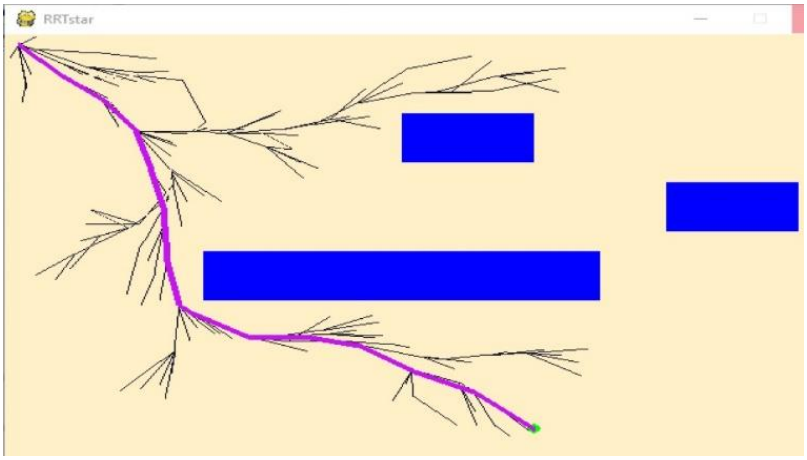

Fig. 6. The growth of tree using RRT* fully optimized, goal bias and aggressive growth, number of computations =222

D. Simulation Model

To validate the model simulations were performed with the following parameter values.

- Dimensions of workspace 500 cm X 500cm
- Number of obstacles 3
- Number of grids 100
- Size of grids 50 cm X 50 cm
- Number of iterations 1000
- Neighborhood range 50

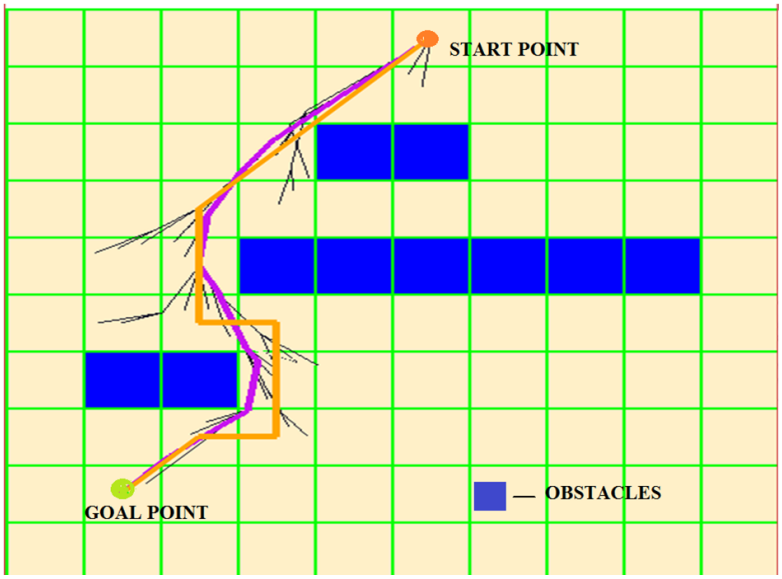

Fig. 7. Simulation result for Grid Navigation

IV. QUEUEING THEORY AND SCHEDULING

In this paper we consider a restaurant setting where customers wait for service by the robot and the robot queues the requests and serves them on a first come first serve basis. We assume a Poisson stream of arrivals which corresponds to arrivals at random with inter arrival times exponentially distributed. The service times for customers are independent. Another common assumption about service times is that they are exponentially distributed.

We have generated the requests from 4 customers. The arrivals are modelled using Poison distribution and the inter arrival times between the subsequent requests follow an exponential distribution. The requests are entered in a queue and serviced in a first come first serve basis, as long as the requests are not simultaneous. When more than one request arrives within a span of 2 seconds, it is considered as a simultaneous request and the priorities of the requests are checked.

The priority of a customer is ascertained by a number of factors such as the distance, number of obstacles, frequency of request etc. Weights are assigned to the different parameters and the sum total is found for both the simultaneous requests. The request that gets the highest weight is chosen for execution immediately. After the first request, generation of subsequent requests and the computation of path for previously received requests is done in parallel.Once the path to service a request is found, the next request is dequeued and taken for execution.

V. LOCALIZATION AND NAVIGATION

A. Indoor localization

Localization is a fundamental problem in robotics. Location information is essential for planning and decision making processes. If a robot does not know where it is, it can be difficult to determine what to do next. IR sensors are used to provide the robot with access to relative and correct measurements giving the robot feedback about its driving actions and the situation of the surrounding of the robot. What makes this difficult is the existence of uncertainty in both the driving and the sensing of the robot. The robot uses odometry for navigation and to obtain its position. The IR sensors count the number of revolutions that the wheels make while driving and turning. This helps us calculate the distance travelled by the wheel from the initial position and hence the robot's current position and orientation is also obtained. Given this information, the robot's location can be determined. However, due to wheel slippage or other small error sources, the odometer readings may provide incorrect results.

Suppose our robot is at (x, y, θ), with θ representing the vehicle's then,by knowing how much the left and right motors have rotated and the diameter of the wheel we can compute the actual distance that the wheel has moved.

The distance moved by the left and right wheels ($d_{left}$and $_{ht}$ respectively)is obtained as follows

$$dleft = 2 * \pi * R * (left\_tick)/N \quad (1)$$

$$drig\not t = 2 * \pi * R * (right\_tick) / N \quad (2)$$

Where, R = 3.66 (radius of the wheel).

N = 20 (Number of holes in the encoder disc).

After our vehicle moves by d_{left} and d_{right}, we compute the new position(x_{new}, y_{new}, θ_{new}). The relations are as shown

Where

$$\varphi = \frac{\text{dleft} - \text{dright}}{dbaseline} \quad (3)$$

$$\theta new = \theta old + \varphi \quad (4)$$

$$xnew = xold + dcenter * cos\,\theta \quad (5)$$

$$ynew = yold + dcenter * sin\,\theta \quad (6)$$

Odometry can provide good dead reckoning over short distances, but error accumulates very rapidly. The error (noise) will be injected into θ also not only in x and y at every step. The error in θ will be maximized in the following iterations. There are a number of basic error sources, which are enlisted below

- Sensor error: If anything other than quadrature phase encoders are or feedback, then the estimates of the d_{left} and d_{right} will be noisy.
- Slippage: While turning on the corner, the robot's one wheel slip a little bit. The robot will not be recoverable even if the odometry data is perfect.
- Error in estimate of $dbaseline$or in wheel diameters. The odometry result will have a consistent veer in one direction or the other. In fact, often the wheels on the robot may be of slightly different sizes. Small errors may cause large navigation problems.

B. Robot Navigation

Robot navigation is the main work of an autonomous robot to move safely from one place to another. The general problem of navigation can be formulated in terms of the following.

- The robot has to know where it is in order to make useful decisions.
- In order to fulfill a task, the robot has to know where it is going.
- Once the robot knows where it is and where it has to go it has to decide on how to get there.

In our implementation, the heading of the robot is obtained from odometry. Given a goal to the robot, the optimized RRT* algorithm is used to find the path to the goal. Once the complete path is obtained from the optimized RRT*algorithm, the intermediate goal points in the path are determined and the appropriate wheel velocities need to be given for moving the robot from its current location to the first intermediate goal point.

This procedure is performed for all the intermediate goal points until the main goal point is reached. This is implemented in two steps, which are enlisted below;

1)In-Place Rotation: In order to facilitate the movement of the robot from one grid to another, the robot is rotated in place at every grid so that its orientation matches the orientation required for traversal from the current grid to the next intermediate goal point.

Algorithm 1 In-Place Rotation

1 **Yrad**: Yaw angle of the robot in radian.

2 Compute θ as

$$\theta = \frac{\text{ygoal} - \text{ycurrent}}{\text{xgoal} - \text{xcurrent}}$$

3 **if** absolute (Yrad - θ) <π then

4 **if** (Yrad<θ) then

5 rotate right until Yrad = θ

6else

7 rotate left until Yrad =

8 **if** absolute (Yrad - θ) >π then

9 **if** (Yrad<θ) then

10 rotate left until Yrad = θ

11else

12rotate right until Yrad = θ

*2)Moving from one block to another:*In this step, the Euclidean distance between the robot's initial position and the final position, in terms of pixels is determined. This distance is converted to actual distance on the floor by multiplying it with a scaling factor. Then, the number of rotations of the wheel that would be required to traverse this distance is then computed and correspondingly, the required number of ticks from the left and right wheel are determined. Both the left and right motors are made to rotate forward until the total number of ticks equals the desired number of ticks. For every incremental tick, odometry function is invoked so that the robot's position is up to date.

Algorithm 2 Moving from one block to another

1 $Euclidean\ distance = \sqrt{(ygoal - ycurrent)2 + (xgoal - xcurrent)2}$

2 Actual Distance = Euclidean distance * 0.7

3 Rotations needed = actual distance / (2* π * R)

4 Total Ticks needed = rotations needed * 20

5 start left motor, right motor

6 **while** (left_ticks< Total Ticks **and** right_ticks< TotalTicks)

then

7 Move Forward

8 Update Odometry

9**stop left motor, right motor.**

VI. HARDWARE IMPLEMENTATION

A. Raspberry Pi

The Raspberry Pi is a credit-card sized computer that uses an ARM cortex A7 Broadcom processor. The Raspberry pi 2 consists of a Broadcom BCM2836 Soc ARM cortex A7 processor with a 900MHz range. The raspberry pi uses a raspian based Linux operating system. It can be used along with peripherals such as keyboard, mouse and monitor. It is a fully featured micro- computer squashed onto a circuit board measuring approximately 9cm x 5.5cm. The OS is uploadedto a SDcard which serves as the memory for the processor. The General purpose input/output pins are a physical interface between the Pi and the outside world. At the simplest level, they can be considered as switches that can be turned on or off (input) by the user.

B. Wheel Encoders

The Optical encoder consists of two components.

1) Encoder disk: The optical encoder disc is a thin cylindrical disc made of glass or plastic with transparent and opaque areas. This is mounted on to the shaft and rotates along with the wheel.

2)IR transceiver: A light source and photo detector array reads the optical pattern that results from the disc's position at any one time. This code can be read by a controlling device, such as a microprocessor or Micro controller to determine the angle of the shaft. When light emitted by the IR LED is blocked because of the alternatings lots of the encoder disc (also known as index disc), conduction level of the photo transistor/diode changes.

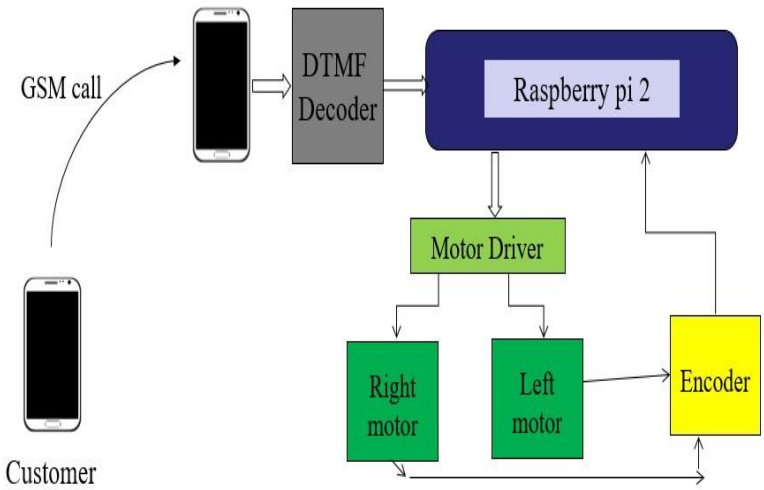

Fig. 8. Hardware implementation of the autonomous robot.

C. Motor Drivers

Motor drivers generate the timing signals for the motors.

D. Dual Toned-Multi Frequency Communication (DTMF)

DTMF is for sending requests from the customers to the robot. The touch phone is a DTMF generator that produces DTMF tones as the buttons are pressed. The circuit generates a unique DTMF tones when the buttons in keypad of the mobile phone are pressed. At the robot the DTMF Decoder acts as band-split filter and determines the key pressed. The output is produced in a 4-bit binary format. When the buttons are pressed on the keypad, a connection is made that generates two tones at the same time-a 'ROW' tone and a 'COLUMN' tone. These two tones identify the key pressed to any equipment which is controlled. When digit 1 is pressed on the keyboard, we generate the tones 1209 Hz and 697 Hz.

1	2	3	697 Hz
4	5	6	770 Hz
7	8	9	852 Hz
*	0	#	941 Hz
1209 Hz	1336 Hz	1477 Hz	

Fig. 9. DTMF Low and High frequency tones and decoded output.

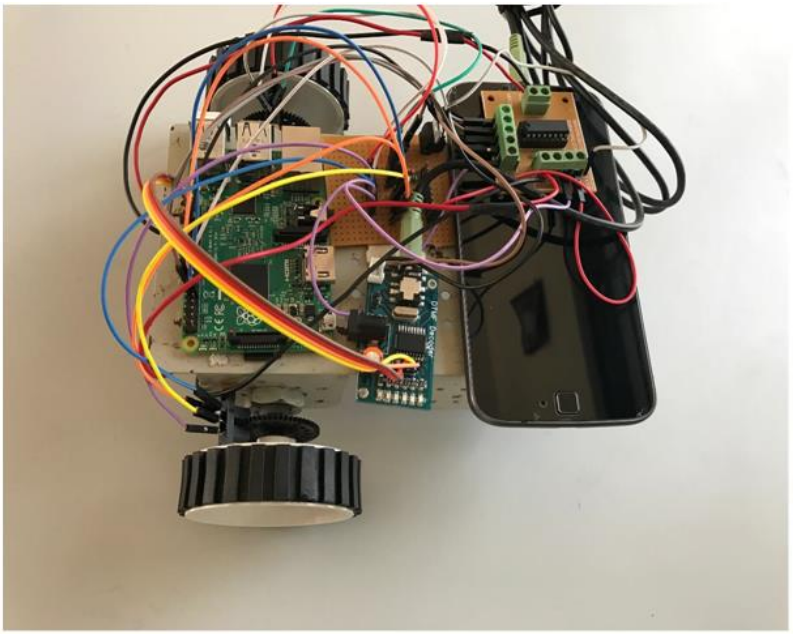

Fig. 10. Autonomous indoor navigating robot.

VII. RESULTS AND DISCUSSION

The autonomous indoor navigation robot provides one such application of a mobile robot which is capable of navigation in an Office/Restaurant/Domestic service environment to fulfill the needs of the user. The improvements in RRT and RRT* made, shows a significant increase in performance than the previous algorithms while computing the time taken to obtain the shortest path. Thus, the algorithm has been successfully modified to be time optimized. Fig 11. Shows the developed algorithm is proven to reduce the computational time to find the shortest path and assures shorter queuing time/waiting time for customers.Fig 12. Shows the performance of the optimized RRT* is found to be three times better than the original algorithm

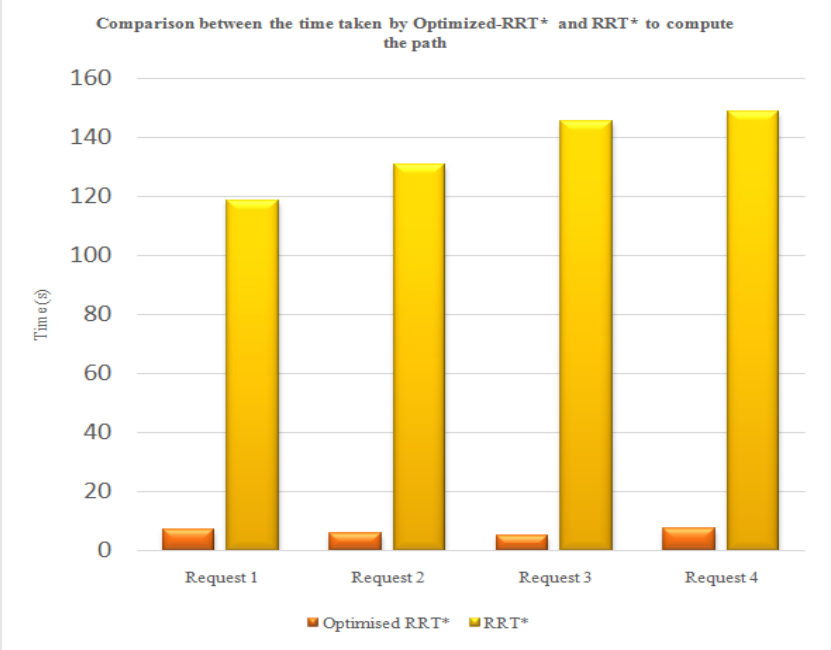

Fig. 11. Comparison of RRT* and RRT*(Optimized)

TABLE I. Analysis of RRT* Algorithm

Request Coordinates (X, Y)	**Time taken to dequeue the request (seconds)**	**Time taken to find the path using RRT* (seconds)**	**Time taken by Raspberry Pi to complete the computations for the request from the time of arrival (seconds)**	**Distance Travelled (pixel units)**	**Time taken by the robot to reach the goal (seconds)**	**Total delay in servicing the customer (seconds)**
270, 95	**10**	**118.67**	**128**	**389.848**	**38.9**	**166.9**
180, 380	129	130.88	260	753.27	75.32	335.32
60, 250	287	145.20	401	932.23	93.2	494.2
575, 100	409	148.73	558	1475.34	147.5	705.5
		Average time taken to compute path using RRT*	**135.87**		**Total time taken to service 4 customers**	**1701.92**

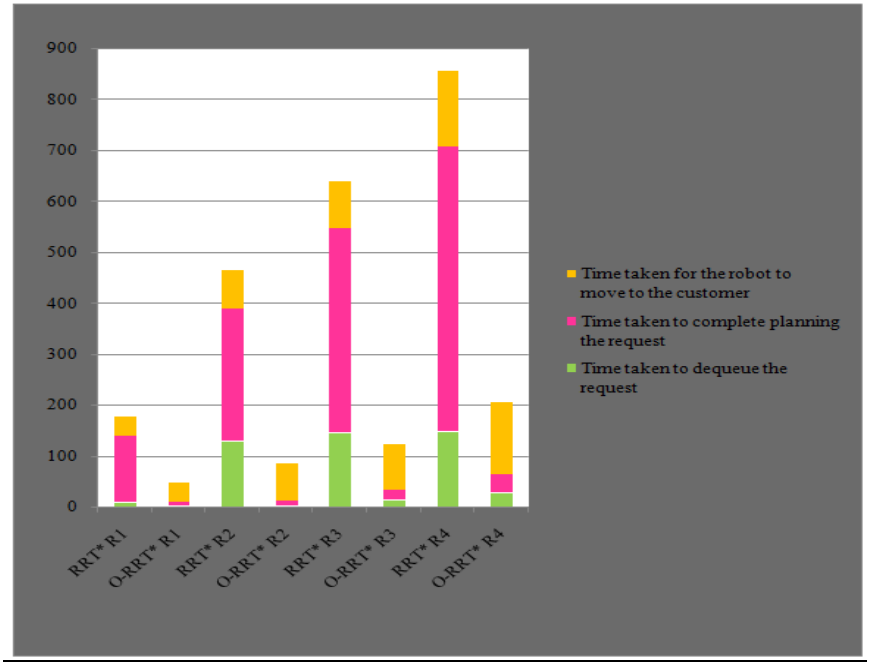

Fig. 12. Comparison of Servicing Time.

TABLE II. Analysis of RRT* Algorithm (Optimized)

Request Coordinates (X, Y)	Time taken to dequeue the request (seconds)	Time taken to find the path using RRT* (seconds)	Time taken by Raspberry Pi to complete the computations for the request from the time of arrival (seconds)	Distance Travelled (pixel units)	Time taken by the robot to reach the goal (seconds)	Total delay in servicing the customer (seconds)
270, 95	**2**	**7.342**	**7**	**391.02**	**39.1**	**46.1**
180, 380	3	5.95	9	731.28	73.1	82.1
60, 250	14	5.16	20	879.80	87.9	107.9
575, 100	28	7.46	35	1419.54	141.9	177
		Average time taken to compute path using optimised RRT*	**6.47**		**Total time taken to service 4 customers**	**413.1**

VIII. CONCLUSION

Autonomous mobile robots in service of mankind are getting closer to becoming reality. The robots at home have the ability to help the handicapped manoeuver and serve as companions for the elderly. There are examples of robots which move around factory environments, domestic households but they are typically guided by bar codes on walls, color lines on the floor or use active beacons. But if a robot could localize itself without any external navigator in its natural environment there will be greater savings on infrastructure, cost, time spent, etc. and highly flexible production or service could be achieved. If the horizon is widened to include all applications for mobile robots, the benefits are even higher of the system that can localize itself using the information from its surroundings. This work can be extended by designing a configurable robotic arm so that the robot can be used at different places such as warehouse, restaurants, library, homes, offices, etc. without much change to the mechanical framework, by enabling the robot to map the environment autonomously using LIDAR or increasing the number of robots and establish communication between them so that the robot closest to the user can service the request rather than a robot that is far away.

REFERENCES

[1] Amritanshu Srivastava, Shubham Vijay, Alka Negi, PrasunShrivastava and Akash Singh, “DTMF Based Intelligent Farming Robotic Vehicle”, Proc.of International Conference on Embedded Systems, Coimbatore, India, pp. 206-210, 2014.
[2] Dong JinSeo and Jongwoo Kim, “Development of Autonomous Navigation System for an Indoor Service Robot Application”, Proc. Of IEEE International Conference on Control, Automation and Systems, Gwangju, Korea, pp. 204-206,2013.
[3] James J. Kuffner Jr. and Steven M. LaValle, “RRT Connect: An Efficient Approach to Single-Query Path Planning”, Proc. of IEEE International Conference on Robotics and Automation, USA, pp. 995-1001, 2000.
[4] Jiadong Li, Shirong Liu, Botao Zhang and Xiao Dan Zhao “RRT-A* Motion Planning Algorithm for non-holonomic Mobile Robot”, Proc. of Annual Conference of Society of Instrument and Control Engineers, Japan, 2014.
[5] Johann Borenstein and Liqiang Feng “Measurements and Correction of Systematic Odometry Errors in Mobile Robots”, IEEE Transactions on Robotics and Automation, Vol. 12, No. 6, pp. 869-880,2010.
[6] Karaman and Frazzoli “Incremental Sampling based Algorithms for Optimal Motion Planning”, The International Journal of Robotics Research Vol. 30, No. 7, pp. 34-41, 2011.
[7] Liang Ma, JianruXue, et al “Efficient Sampling based Motion Planning for On Road Autonomous Driving”, IEEE Transaction on Intelligent Transportation systems, Vol. 16, pp. 1961-1976,2015.
[8] Liang Ma, JianruXue, Kuniaki Kawabata, Jihua Zhu, Chao Ma and Nanning Zheng “A Fast RRT Algorithm for Motion Planning of Autonomous Road Vehicles”, Proc. of IEEE International Conference on Intelligent Transportation Systems (ITSC), China, pp. 1033-1038,2014.
[9] Sunhong Park and Shuji Hashimoto “Autonomous Mobile Robot Navigation Using Passive RFID in Indoor Environment”, IEEE transactions on Industrial Electronics, Vol. 56, pp. 2366-2373,2009.

AUTHOR PROFILE

Dr. R. Jayaparvathy, Professor, Department of ECE. She obtained her Ph.D in Information and Communication Engineering from AU-KBC Research Centre, Anna University, Chennai. Her areas of interest include Wireless MAC, Wireless Sensor Networks including Body Area Networks and Embedded Systems

Sheeba Angel A, Research Scholar, Department of ECE. She obtained her PG degree in VLSI Design with distinction from SSN College of Engineering, Chennai. Her areas of interest include embedded systems and modelling.

Nitin Barattwaj P, Department of ECE, SSN College of Engineering, Chennai.

SrihaarikaVijjappu, Department of ECE, SSN College of Engineering, Chennai.

Srisruthi sridhar, Department of ECE, SSN College of Engineering, Chennai.