Path Planning and Parking Assistance: Informed RRT with non-Holonomic System

Linfeng Li, Shubham Wani, Yang Liu

***Abstract*—This report presents the development and implementation of a path-planning algorithm for robotic vehicles and also develops and implements a robotic parking assistant system designed to guide a vehicle into a parking spot autonomously. The algorithm is based on Rapidly-Exploring Random Trees (RRTs) and is designed to find the shortest path from an initial position to a goal position. Then several optimization methods are applied to improve the convergence rate and accuracy. Our project is implemented in a simulated environment, allowing us to evaluate the algorithm's performance and identify improvements that could be made. This report outlines the design and implementation of the algorithm, discusses the challenges encountered, and provides conclusions and recommendations for further development.**

Keywords—path planning, RRT, Optimization

I. INTRODUCTION

Robots and AI play a significant role in our lives. They can take care of tedious or hazardous tasks, allowing people to focus on more complex work or leisure activities. In this project, we developed path planning and parking assistance technology, which has the potential to enhance safety in our daily lives. As self-driving vehicles become more prevalent, parking assistance can reduce the risk of accidents by providing a dependable and precise method for robots to guide vehicles into parking spaces. It can also decrease traffic congestion in urban areas by enabling vehicles to park more efficiently and quickly. Additionally, parking assistance can decrease fuel consumption by allowing robots to guide vehicles accurately to the most appropriate parking spots.

II.RRT

Steven M Lavalle first developed Rapidly Exploring Random Trees at Iowa State University. They were considered novel for their ability to explore the entirety of the configuration space and their ability to handle differential constraints. They served as the baseline for multiple improvements on RRT, such as RRT*, Fixed Node - RRT, and Improved RRT.

RRT works by sampling a uniformly random point in the configuration space and finding the closest node in the graph structure(tree) based on the lowest cost function value. Euclidean distance $\sqrt{(Xgoal - Xrand)^2 + (Ygoal - Yrand)^2}$ is generally used as a cost function for 2D planning problems. The graph grows from the closest node towards the random point unless an obstacle is encountered. The growth of the graph accounts for the differential constraints for non-holonomic robots such as cars and drones.

MATLAB R2021a was used to create simulations for path planning. A *100x100* configuration space with one obstacle was defined. This case was different as we needed to store the angular orientation of the car as we propagated the graph(tree) with motion primitives. The number of nodes was limited to *500* for low execution times.

Each new node was stored in memory using *structures* data type, with fields for *self-location number, parent location number,* and *distance.* We generated a random point and grew the closest node towards it, accounting for the differential constraints with the Eulers formula. Bias in the sampling set the random point to the goal point to growing the tree to the goal position for *~5%* iterations. Obstacles were rectangular, so obstacles were avoided using simple constraints on *x1<a(x), x2>a(x), y1<a(y), and y2>a(y).* The graph(tree) grew until one node was populated into the threshold circle around the goal position. When this condition was reached, the nodes were backtracked to generate a kinematically feasible path and shown on a plot. The small lines on the circle show the agent's direction, inferring that the angle does not change discontinuously.

Various enhancements were made to this Vanilla RRT to adapt it to the parallel parking scenario, as we will see in the later sections.

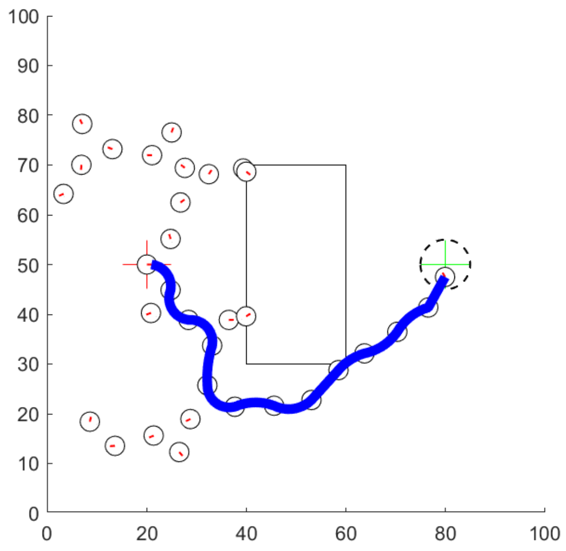

Fig.1. Path Planning with RRT

III. OPTIMIZATION (INFORMED RRT)

The Rapidly-Exploring Random Tree (RRT) algorithm is widely used for exploring a problem space and finding a solution if there is one. However, there are some limitations. The first issue is that the algorithm's convergence rate increases non-linearly as the problem space's complexity increases, making it less efficient in more challenging environments, such as in irregularly-shaped problem spaces with multiple obstacles. Additionally, each step of the algorithm is biased by the existing RRT tree and is unable to improve upon existing nodes, resulting in non-optimal paths that may not be further optimized. After an existing path to the goal is found, the probability of finding alternate paths is biased negatively by the existing path.

To address these issues, the idea of Informed RRT was utilized for our motivation from the article "Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic "[2]. Informed RRT means changing the sampling bias based on our information, which is an extension of RRT. Informed RRT works by adding a sampling bias towards selecting random points within a heuristically determined ellipsoid centered on the start and goal nodes. This sampling bias is only utilized once an initial valid path between the start and goal nodes has been found, allowing informed RRT to explore the problem space and find alternate paths more efficiently.

The use of the ellipsoid and the sampling bias towards points within it allows informed RRT to quickly find alternate paths and improve upon existing solutions, particularly in complex problem spaces. In addition to addressing some of the limitations of RRT, this extension also offers the potential for improved efficiency and performance in a variety of applications.

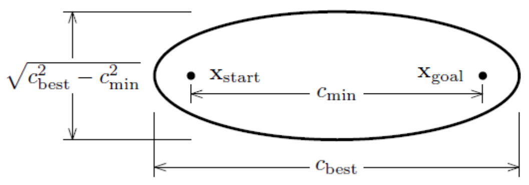

Fig.2. The Heuristic Sampling Domain[2]

We can consider the set of possible states in a problem as X, where $X \subseteq R^n$. The set of states that are found through heuristic sampling can be represented as Xf, $and\ X_f \subseteq X$. [2] Using an ellipse as a sampling domain leads to better results because points outside of the ellipse are not included in the data set. To create the ellipse, we need to specify an initial state as x_{start} and a goal state x_{goal}. The minimum cost between these two focal points c_{min} and the cost of the current best solution c_{best} can then be used to determine the shape of the ellipse and its eccentricity, as c_{min}/c_{best}[2]. It is important to note that the Euclidean distance between two points is an acceptable heuristic for this problem, given the non-holonomic constraints. Assuming that the improved subset of state space is X_f, the closed form function can be expressed in terms of x_{start}, x_{goal} and c_{best} as following,

$$X_f = \{\mathbf{x} \in X \mid \|\mathbf{x}_{start} - \mathbf{x}\|_2 + \|\mathbf{x} - \mathbf{x}_{goal}\|_2 \leq c_{best}\}$$

[2]

The conjugate diameters can also be expressed as $\sqrt{c_{best}^{\ 2} - c_{min}^{\ 2}}$ (fig. 2).

```
if $c_{\max} < \infty$ then
    $c_{\min} \leftarrow \|\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{start}}\|_2$;
    $\mathbf{x}_{\text{centre}} \leftarrow (\mathbf{x}_{\text{start}} + \mathbf{x}_{\text{goal}})/2$;
    $\mathbf{C} \leftarrow \texttt{RotationToWorldFrame}(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}})$;
    $r_1 \leftarrow c_{\max}/2$;
    $\{r_i\}_{i=2,\ldots,n} \leftarrow \left(\sqrt{c_{\max}^2 - c_{\min}^2}\right)/2$;
    $\mathbf{L} \leftarrow \text{diag}\{r_1, r_2, \ldots, r_n\}$;
    $\mathbf{x}_{\text{ball}} \leftarrow \texttt{SampleUnitNBall}$;
    $\mathbf{x}_{\text{rand}} \leftarrow (\mathbf{CL}\mathbf{x}_{\text{ball}} + \mathbf{x}_{\text{centre}}) \cap X$;
else
    $\mathbf{x}_{\text{rand}} \sim \mathcal{U}(X)$;
return $\mathbf{x}_{\text{rand}}$;
```

Fig.3. Informed RRT Algorithm (image taken from the paper "Informed RRT: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic")[2]

The algorithm described above shows how to generate a heuristic sampling domain using code, without considering non-holonomic system constraints. To further understand how the algorithm works, it is helpful to know that it returns samples from an ellipse and calculates the best solution cost between two focal points within C_{max} [2].

IV. PARKING IMPLEMENTATION: parallel parking assistance is implemented after finding the path from the start position to the goal with the lowest cost based on the informed RRT with the non-Holonomic system.

To define the map, several obstacles were set to build the parallel parking environment. The modeling of the car structure is simplified to a rectangular shape with realistic dimensions, where the length of the car is about 2.5 times the width.

To mathematically model the car, we assume the car is defined if we know the coordinates of the four corners of the rectangle. And these can be derived based on the coordinate of the car's center, the width and length of the car, and the car's driving angle. The following equations are used:

$$\begin{cases} x_1 = x_c - w\cos\gamma - b\sin\gamma \\ x_2 = x_c + w\cos\gamma - b\sin\gamma \\ x_3 = x_c + w\cos\gamma + b\sin\gamma \\ x_4 = x_c - w\cos\gamma + b\sin\gamma \end{cases}, \quad \begin{cases} y_1 = y_c - w\sin\gamma + b\cos\gamma \\ y_2 = y_c + w\sin\gamma + b\cos\gamma \\ y_3 = y_c + w\sin\gamma - b\cos\gamma \\ y_4 = y_c - w\sin\gamma - b\cos\gamma \end{cases}$$

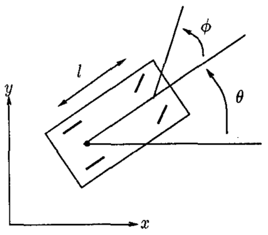

Fig. 4. Mathematical Expression for Cars[4]

The goal is to come up with a parking solution automatically. The car is defined to be successfully parked as long as the four wheels are inside the parking structure. Also, a collision check is implemented using the poly shape overlap check in MATLAB, where the car should not be overlapping with any obstacle. The graph below shows the parking structure and the trace of the car when it has parked successfully.

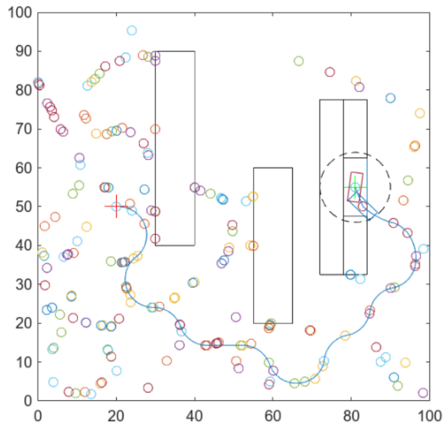

Fig.5. Modeling of Obstacles and Car

V. NEAR-GOAL SAMPLING AND OPTIMIZATIONS

To optimize the parking structure to increase its efficiency, reduce the convergence time, and increase accuracy, some optimization is done. Since RRT is a process where random points are generated in the whole space randomly, it is a good way for path planning from two points far away from each other. However, it is not so suitable for parking. When parking, the path for the car needs to be much more accurate.

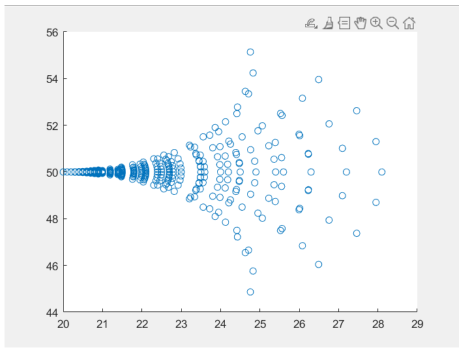

Fig.6. Motion Primitives

The circles in figure (X) represent all the possible endpoints of a path starting from the parent node on the left. These endpoints are calculated according to our motion primaries, with each combination of the velocity and steering angle resulting in each endpoint on the plot. The motion primitives can be modified to be more accurate and smooth near the parking structure if the following changes are made:

1. Higher precision: set smaller step sizes for velocity and steering angle.
2. Reverse gear allowed: linear velocity range changed from all positive into negative to positive.
3. Wider steering angle range.
4. Once any path is within a threshold of the parking spot, add a large goal bias of 50% with random sampling within the threshold, and at the same time, continue RRT to search for new paths.

After doing these optimizations, the motion primitive will be much more smooth; in the meantime, whenever the car is near the parking structure, it has half of the chance to stay in the parking structure and sample more points in that area, so it would be much more heavily sampled in the parking structure area so that the parking is done more accurately. And below is a graph showing that the nodes of RRT are heavily generated within the parking structure area, which meets our requirement.

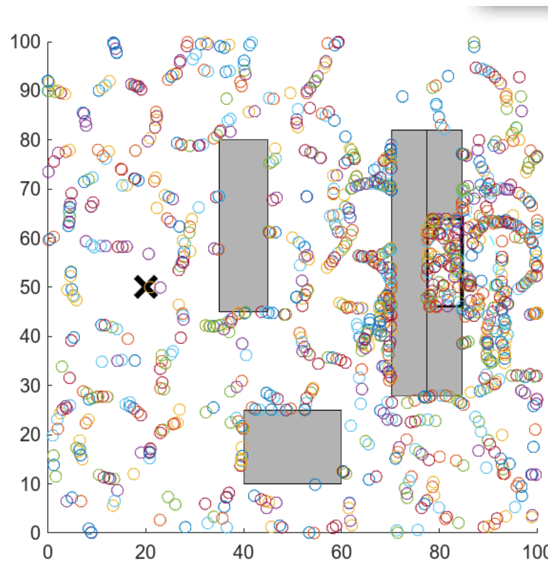

Fig.7. Goal-Sampling Example

VI. RESULTS AND BENCHMARKING

Using our optimization methods, the program converges faster, and there is a benchmarking of three cases where we compare the computation times and total nodes generated to find a path planning and parking solution. The total nodes generated should be able to represent the computation time.

Case 1: With Goal Sampling Optimization only

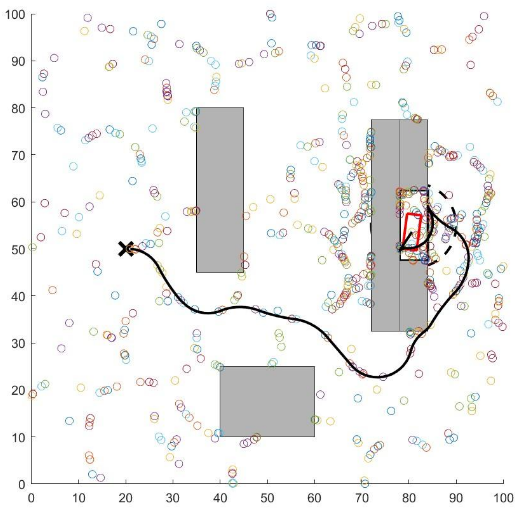

Fig.8. Example Result of Goal Sampling Optimization

Total nodes generated: **N=782**

Case 2: With Ellipse Informed RRT and Goal Sampling

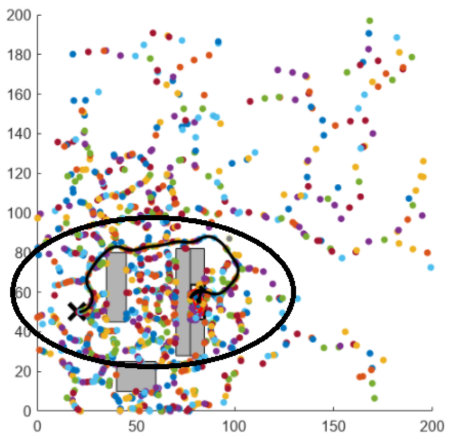

Fig.9. Example result of Goal Sampling Optimization with Informed RRT

Total nodes generated: **N=489**

Case 3: Without Goal Sampling:

Almost impossible to converge, **N>2000**

From this it is clear that with goal sampling and ellipse-informed RRT optimizations, it took fewer nodes generated to succeed in parking, without any optimization, it was almost impossible to converge within a reasonable time.

Finally, we chose to use ellipse-informed RRT combined with goal sampling optimization, finish running a total number of 2000 nodes, save all possible paths, and only output one path with the lowest cost function. That path is considered the optimal path. The cost function is defined as *cost(node) = cost(parent_node) + distance,* which represents the total distance from each node to the start point.

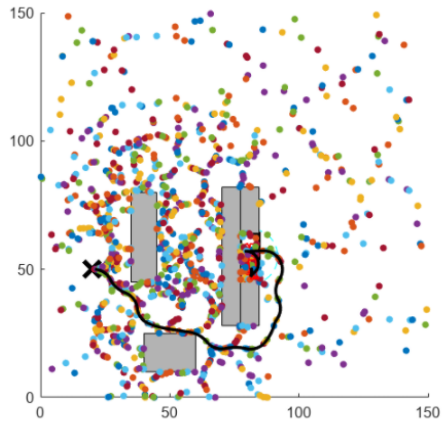

Fig.10. Example of Final Result

In this case, after generating 2000 nodes, four paths were found, and the shortest path was plotted.

VII. CONCLUSION

In this project, we implemented optimal path planning and parking assistance. Our system was tested in a simulated environment and was found to be successful at guiding vehicles into parking spots and finding the shortest path to a destination. Ellipse-Informed RRT and goal-sampling optimizations were used to improve the efficiency and accuracy of the system. The results demonstrate the potential of this technology to improve the safety, efficiency, and cost-effectiveness of parking structures and autonomous vehicles.

VIII. REFERENCES

[1] Rapidly-Exploring Random Trees: A New Tool for Path Planning, Steven L Lavalle.
[2] J. D. Gammell, S. S. Srinivasa and T. D. Barfoot, "Informed RRT: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 2997-3004, doi: 10.1109/IROS.2014.6942976.
[3] G. Wahba, “A least squares estimate of satellite attitude,” SIAM Review, 7: 409, 1965.
[4]Welcome to caltechauthors. Welcome to CaltechAUTHORS - CaltechAUTHORS. (n.d.). Retrieved December 3, 2022, from