Updated Wavefront Planning Algorithm for Ackerman Steering System and its Comparision to Sampling-based planner RRT.

Algorithmic Motion Planning Final Project

Kedar C. More
Aerospace Engineering
University of Colorado, Boulder
Boulder, U.S.A
kemo6996@colorado.edu

***Abstract*—This paper introduces an updated version of the existing Wavefront algorithm with path-smoothing for Ackerman Steering System. Most of the autonomous vehicular systems have Ackerman Steering Mechanism. So, most of the research is done on this mechanism. The proposed algorithm is complete as opposed to the sampling algorithms which are probabilistic complete. This paper discusses the advantages of grid-based planner over sampling-based for the calculation of a fast and optimal path.**

Keywords—Wavefront algorithm, RRT (Rapidly exploring Random Trees), Ackerman Steering System, Informed Trees.

I. INTRODUCTION

A. *Ackerman Steering Mechanism*

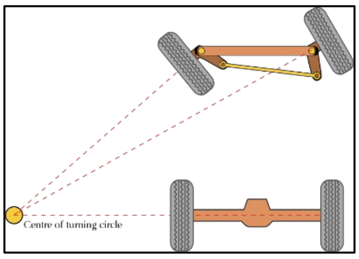


Figure. 1: Ackerman Steering

Ackerman mechanism [1] is used in most of the cars as it come the closest to the ideal steering model. Ideal steering model ensures that the wheels do not experience any lateral friction at any steering angle.

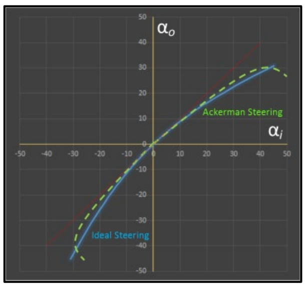


Figure. 2: Ideal Steering vs Ackerman Steering [5]

Here α_0 and α_i are the actual angles of the wheels. After 30 degrees of turn, the Ackerman system start deviating from the ideal steering. But in practical situations this angle is more than enough for the car.

Geometry: It is a four-bar link mechanism with on link as ground and two of the links connected to the ground have equal lengths. These links have the wheels turning with them at the same angle. The remaining link is used as the control input from the steering wheel and is connected to the rack and pinion.

B. *Sampling-based Algorithms*

Most common Sampling-based Algorithm is RRT (Rapidly Exploring Random Tree) [2] and its variants like RRT*, RRT-connect. After RRT many algorithms were developed owing to the ease of their usage and low computation required to run them. Informed Trees are the latest addition to these advancements and are used widely.

```
GENERATE_RRT($x_{init}, K, \Delta t$)
1  $\mathcal{T}$.init($x_{init}$);
2  for $k = 1$ to $K$ do
3      $x_{rand} \leftarrow$ RANDOM_STATE();
4      $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x_{rand}, \mathcal{T}$);
5      $u \leftarrow$ SELECT_INPUT($x_{rand}, x_{near}$);
6      $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
7      $\mathcal{T}$.add_vertex($x_{new}$);
8      $\mathcal{T}$.add_edge($x_{near}, x_{new}, u$);
9  Return $\mathcal{T}$
```

Figure. 3: RRT Algorithm

First, we initialize the tree τ with the start point as a node. Then a valid random point is selected in the workspace as x_{rand}. The node that is nearest to the point is selected as x_{near}. A control input u is selected which will take the robot closest to x_{rand} from x_{near}. The state at which the robot reaches is termed as x_{new}. A new vertex x_{new} is added to τ and also an edge (x_{near}, x_{new}) with the control input u. This process is run K times so that the tree grows out from the start node.

C. *Kinematic Planner*

The difference between normal path planning and kinematic path planning is that the actual path depends on the control inputs available to the robot. Choosing a state does not imply that the state is achievable. This heavily affects the actual path length and the predicted one.

II. WAVEFRONT PLANNER

For this paper we will consider tree workspaces for comparison and benchmarking.

These are the configuration spaces of some Workspaces in 2D.

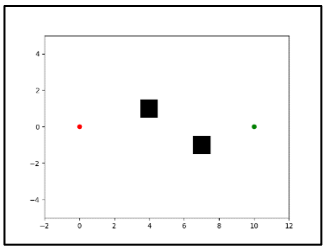


Figure 3: Workspace 1

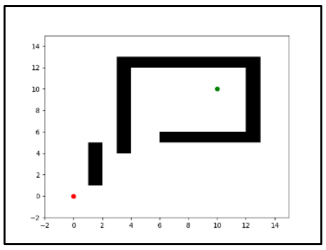


Figure 4: Workspace 2

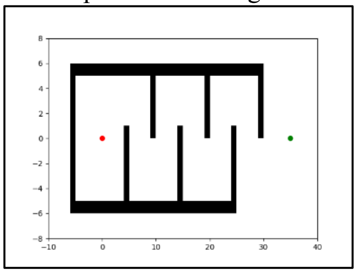


Figure 5: Workspace 3

Here the obstacles are represented by the black space and the white space is the free area where the robot is allowed to move. The red dot is the starting point and the green one is the goal.

The Wavefront Planner [4] is a complete algorithm and is optimal with respect to the grid. It uses a grid-based workspace to calculate a path from start to goal.

First the workspace is divided into grid of a certain size. The values of each grid is updated as follows:

1. Cells with obstacles is assigned the value 1.
2. Assign start cell value to 2.
3. Assign goal cell value to 3.

Update the cell values starting from the goal. Update all the neighboring cells to the value one more than the current cell. Continue this till one of the neighboring cells reaches the start cell with value 2. After that create a path by choosing the neighbors which has a value one less than the current cell. This will automatically reach the goal cell with value 3.The neighbors are described by the cell who share a facet with the other cell.

e.g.

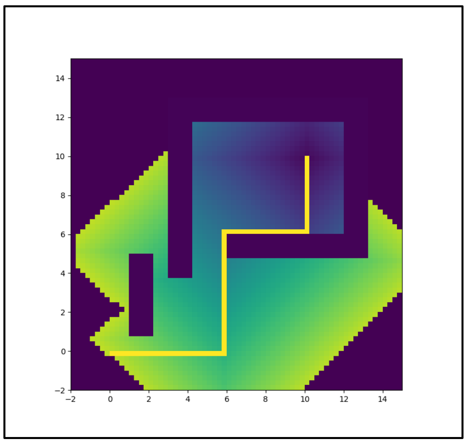


Figure 6: Applying Wavefront Planner on Workspace 2

Here the darker squares (purple) represent the grids with lesser value and in turn the lighter squares (yellow) represent the grids with grids with higher value.

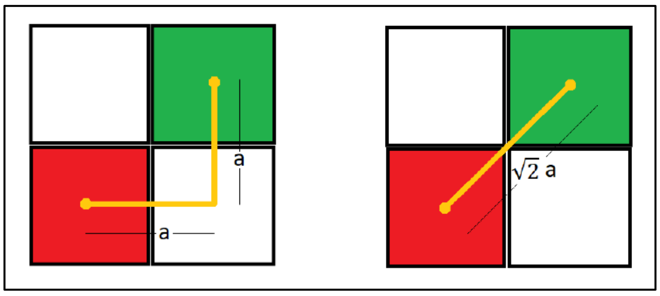
III. Updating the Wavefront Planner

The wavefront planner is already an optimal planner, but it can be improved. In this paper two methods are applied on the existing planner to make it better. The metric for betterment of the planner is considered to be length of the path and the computation time.

The proposed methods are:

1. Changing the definition of Neighbor cells:

Instead of using the neighbor cells as the ones sharing a facet, the cells sharing a facet as well as point are considered as neighbors. This will enable the robot to move to the diagonal cell which can reduce the path length.



(a) (b)

Figure 7: Comparison of path on different kind of neighbors (a) sharing a facet and (b) sharing a facet and point

The grid in this example has the sides of length a. Here the red cell and green cell represents the start and goal respectively. In (a) the total distance of the path is 2*a and in (b) the total distance of the path is $\sqrt{2}$*a. In a best case scenario the final path will be reduced by a factor of $\sqrt{2}$.

2. Choosing the best Neighbor.

In the original planner the neighbors for updating the path are chosen randomly. This might lead to a sub-optimal path for the robot. This method proposes that there can be a cost assigned to each cell and the cell with least cost can chosen for updating the path.

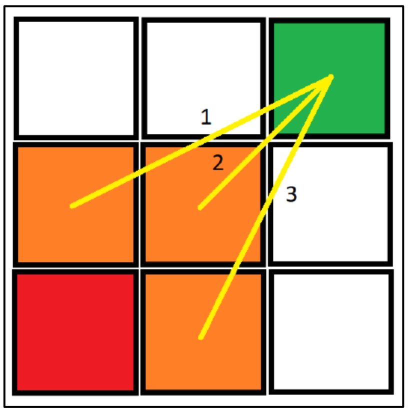


Figure 8: Cost of the cells with respect to the goal

Here the three yellow lines represent the cost of the respective cells. The cost chosen is the Euclidian distance to the goal. Here the Euclidian Distance is defined as the distance between the centers of the cells. We have to choose the cell with the least cost. In this case the path 2 is selected from the start node.

After these updates are applied on the planner, we get the following results for the workspaces.

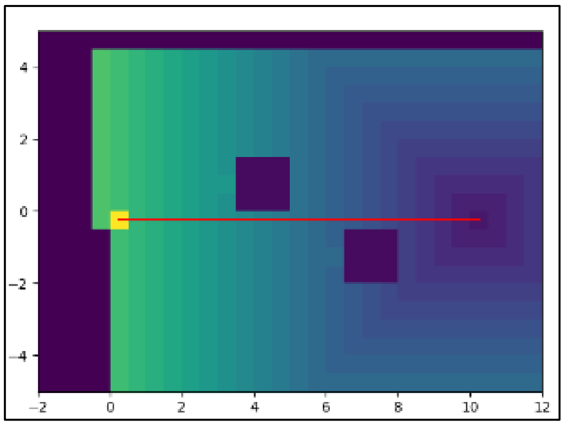


Figure 9: Updated path for Workspace 1

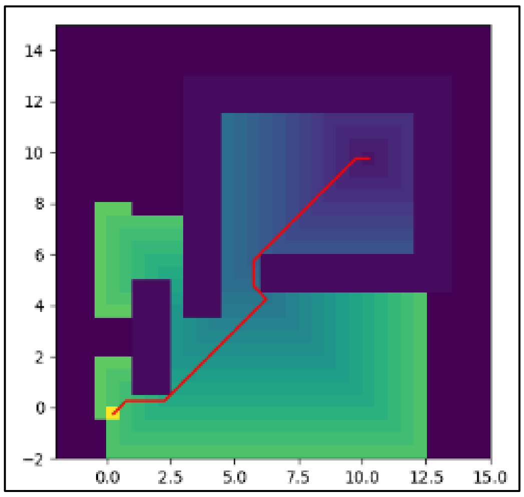


Figure 10: Updated path for Workspace 2

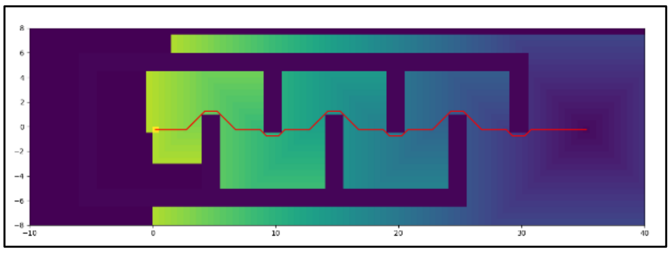


Figure 11: Updated path for Workspace 3

	Workspace 1		Workspace 2		Workspace 3	
	Original	Updated	Original	Updated	Original	Updated
Path Length	10	10	20	15.73	41	39.97
Computation Time (secs)	0.013	0.028	0.036	0.058	0.159	0.304

Table 1: Comparison between the original and updated Wavefront planner

IV. PATH SMOOTHNING

The path we get from the updated Wavefront planner is with respect to the grid, so it has sharp 45 degree turns which might be very difficult to maneuver with the Ackerman steering mechanism. For this a path smoothing technique is applied on the initial discrete path to make it easier for the Ackerman controller to reach the goal.

There are two part for this technique:

1. Eliminate the redundant points on the path.

```
Algorithm 1 Delete Redundant Nodes
Input: list of Nodes (N)
    Output: list of Updated Nodes (UN)
  $UN \leftarrow N[0]$
  $currentNode \leftarrow N[0]$
  while $currentNode \neq N[-1]$ do
      for $i \in N$ do
          if $throughObstacle(currentNode, N[i]) = True$ then
              $currentNode = N[i-1]$
              $UN \leftarrow N[i-1]$
              $break$
          end if
      end for
  end while
  Return: UN
```

Here the main concept used is that we can remove extra nodes if we can reach the next node directly from the previous one. This will reduce the path length by the triangle sides rule. It states that the sum of the two sides of a triangle is always greater than the third side.

2. Introduce curves to the segments of the path. [6]

```
Algorithm 2 Introduce curves to Segments of the Path
Input: list of Nodes (N)
    Output: FinalNodes
  for $i \in N - 1$ do
      $newList \leftarrow (N[i] + N[i+1])/2$
  end for
  for $i \in newList - 2$ do
      $FinalNodes \leftarrow CurvedPath(newList[i], newList[i+1], newList[i+2])$
  end for
  Return: FinalNodes
```

```
Algorithm 3 CurvedPath
Input: Point0, Point1, Point2
    Output: List of points on Path
  $angle1 = slope(Point0, Point1)$
  $angle2 = slope(Point1, Point2)$
  $CurvedPath \leftarrow TwoBoundaryValueProblem(Point0, Point2, angle1, angle2)$
  Return: CurvedPath
```

This will give the list of points on which the Ackerman mechanism can be easily maneuvered. This algorithm makes sure that the turns are continuous and there are no sudden jerks while turning.

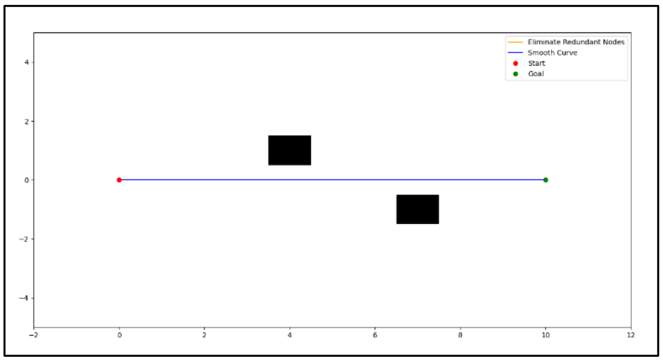


Figure 12: Final Smooth path for Workspace 1

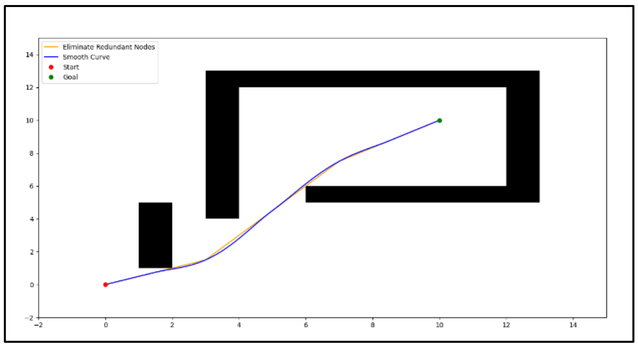


Figure 13: Final Smooth path for Workspace 2

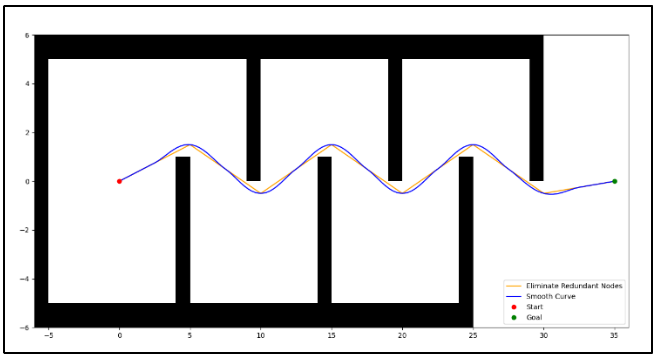


Figure 14: Final Smooth path for Workspace 3

V. Kinematics for Ackerman Steering

Radius of Curvature: [7]

$$R = \left| \frac{\left(1 + y'^2\right)^{\frac{3}{2}}}{y''} \right|$$

Here y is a function with respect to x. R is the list of radii of the curve at every x. This radius enables us to determine the turning radii of the wheels. And as the functions are continuous the turning of the car will be smooth.

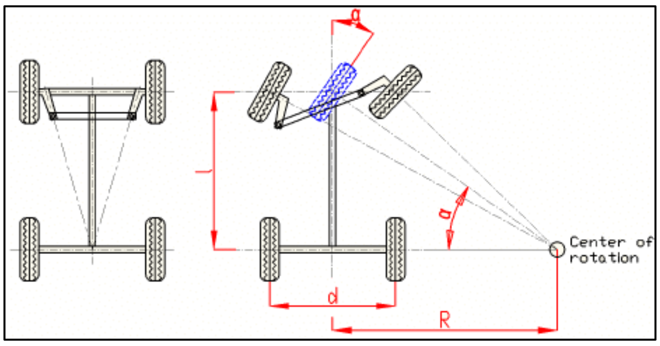


Figure 15: Radius of curvature of the path and the corresponding steering angle.

Here R represents the radius of curvature of the path and α is the angle turned by the wheel in the bicycle model. For the Ackerman steering system, the two front wheels turn at different angles. The car dimensions are considered to be length $= l$ and width $= d$. The following calculations derive the value of the two angles.

$$tan(\alpha) = \frac{l}{R}$$

$$\boldsymbol{\alpha = tan^{-1}\left(\frac{l}{R}\right)}$$

Similarly, for the left wheel:

$$\boldsymbol{\theta_1 = tan^{-1}\left(\frac{l}{R + \frac{d}{2}}\right)}$$

And for the right wheel:

$$\boldsymbol{\theta_2 = tan^{-1}\left(\frac{l}{R - \frac{d}{2}}\right)}$$

The inputs given to the car will be the turning radii and the velocity. To calculate the maximum possible velocity v_m achievable by the car without slipping is by equating the maximum possible friction F_m to the centrifugal force f_c acting on the car. This is due to the directions of these forces being opposite to each other. The friction resists the car to move lateral where the centrifugal force to pushing the car. The centrifugal force on the car depend on the velocity of the car and the radius of curvature of the path. Maximum friction can be calculated by the product of the coefficient of friction μ between the road and the wheels and the weight of the car mg where m is the mass of the car and g is the acceleration due to gravity.

$$F_m = f_c$$
$$\mu(mg) = \frac{m{v_m}^2}{R}$$
$$\boldsymbol{v_m = \sqrt{\mu g R}}$$

The car can be accelerated upto this linear velocity v_m given the coefficient of friction of the ground and the radius of curvature of the road.

VI. Sampling based Methods for Ackerman Steering

The recently developed sampling-based method like AIT* [8] and ABIT* [9] are used on the NASA/JPL-Caltech's Axel Rover System [10] which is a differential drive system. The C-space of the kinematic control system is 2D (input for two wheels). But for Ackerman system using the two value boundary problems additional dimensions are introduced. This cannot be handled by these algorithms efficiently. In other case simple RRT can be used in more than 2D spaces with little or no difficulty.

In the paper [3] RRT is used to plan the Ackerman steering model. It presents Gaussian sampling-based strategy, path formulation meeting system kinematics, and trajectories

creation over the sample points.

Running RRT on the following workspaces give the following results.

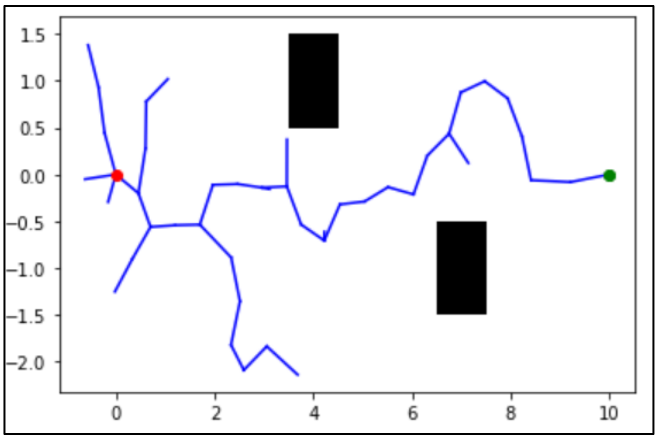


Figure 16: RRT on Workspace1

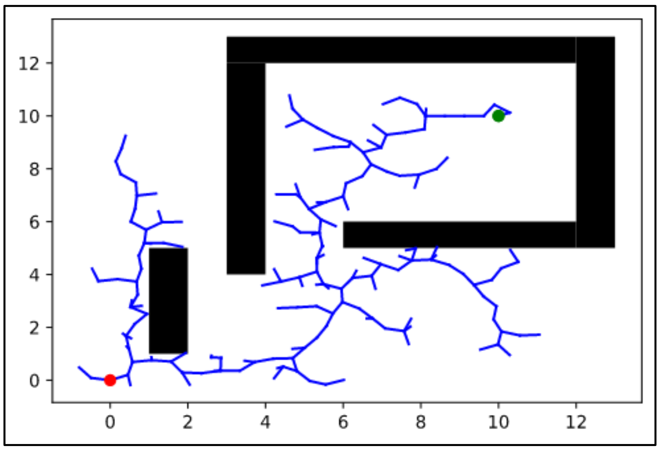


Figure 17: RRT on Workspace2

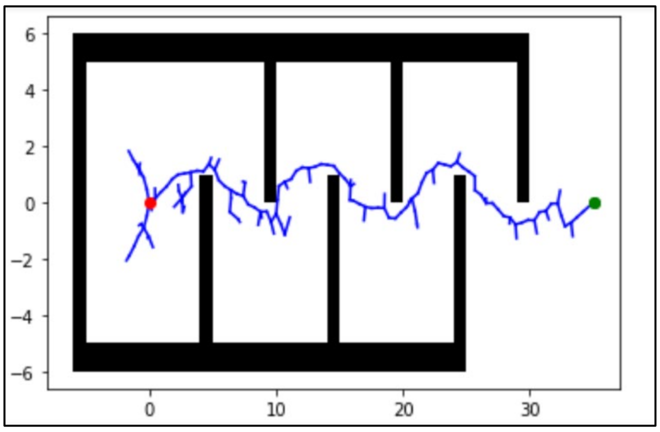


Figure 18: RRT on Workspace3

Kinematic Planning:

Here the Ackerman model is approximated by a unicycle maodel. When we steer to x_{new}, theoretically it gives a straight line form the $x_{nearest}$ to x_{new}. But when we give control inputs with the two value boundary problem, the paths is approximated.

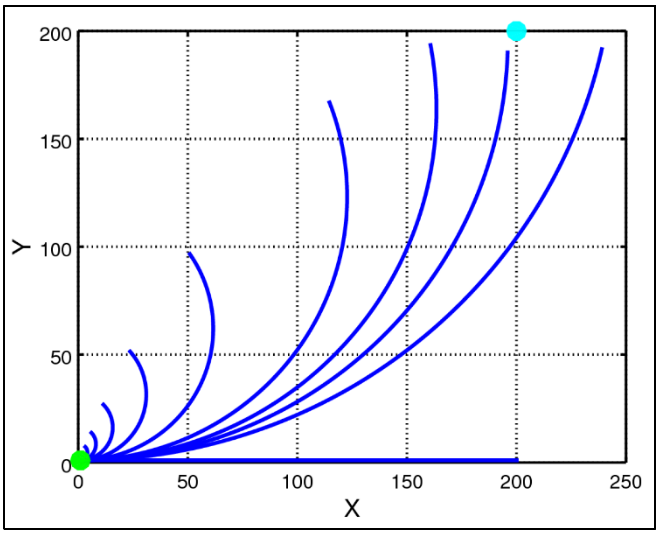


Figure 19: Two Value Boundary Problem (Shooting Method) [11]

Shooting Method is used in this algorithm to solve the two value problem by converting it to single value problem. It is an iterative method using the first order derivative and the end point will reach as close to the next point on the path as possible. This will lead to a smooth path which can be traversed by an Ackerman system. In this paper the back wheel drive system is used where the equations of motion are:

$$\dot{x} = v\,cos(\theta)$$
$$\dot{y} = v\,sin(\theta)$$
$$\dot{\theta} = \frac{v}{L}tan(\delta)$$

Where, θ is the orientation of the vehicle and δ is the input steering angle.

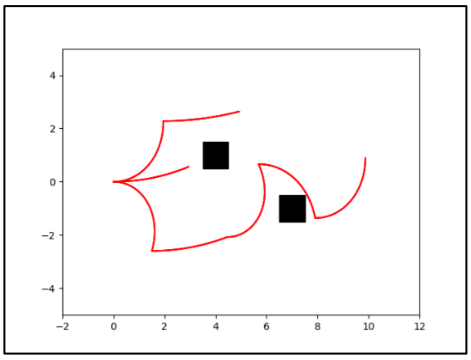


Figure 20: Shooting Method on Workspace1

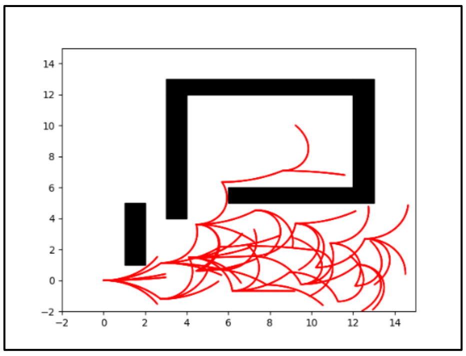


Figure 21: Shooting Method on Workspace2

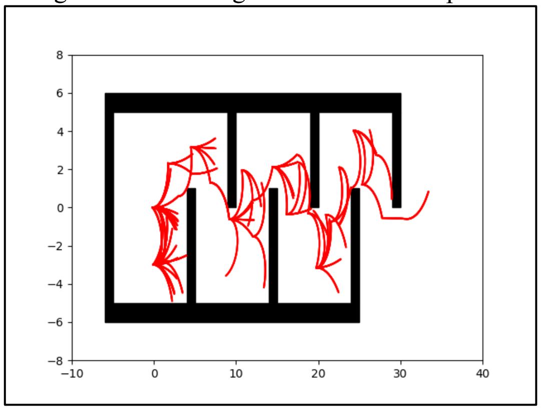


Figure 22: Shooting Method on Workspace3

VII. Benchmarking

For benchmarking the planners were run 100 time and were tested on two parameters:

1. Length of the path
2. Computational time

The benchmarking is performed on the three workspaces.

Updated Wavefront:

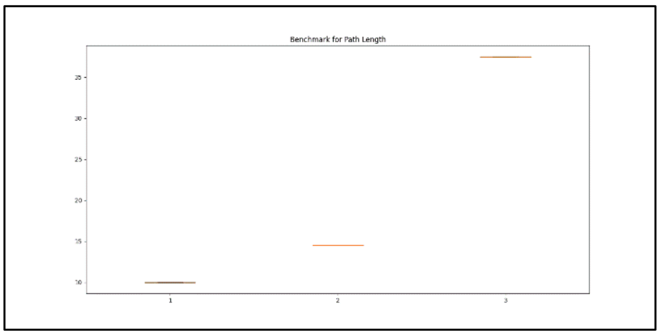


Figure 23: Benchmarking for Path length on Updated Wavefront

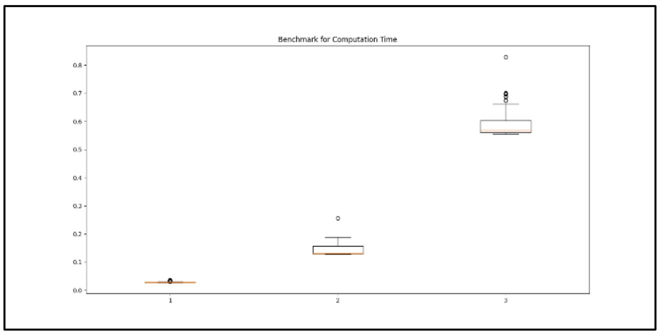


Figure 24: Benchmarking for Computational Time on Updated Wavefront

Here the goal was reached every time in all the workspaces.

[100, 100, 100]

RRT with Shooting Method:

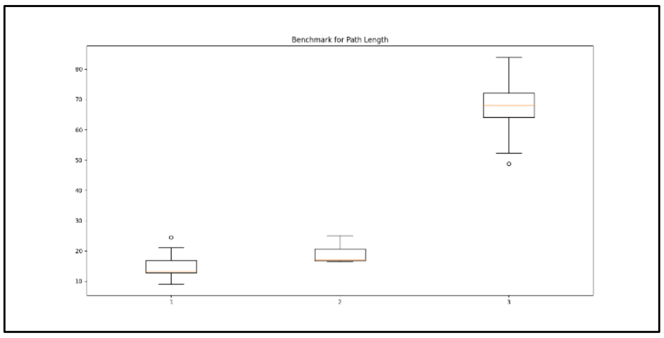


Figure 25: Benchmarking for Path length on RRT

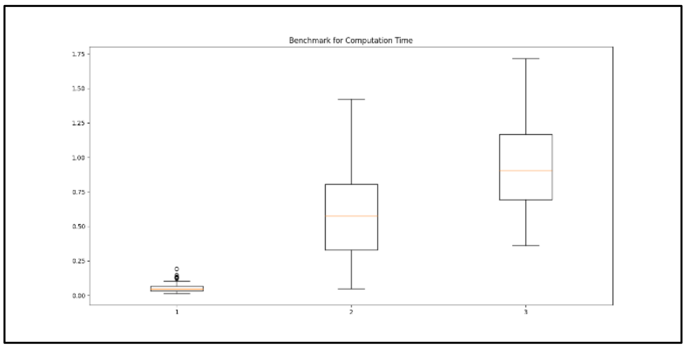


Figure 26: Benchmarking for Computational Time on RRT

Here the goal was reached [100, 62, 62] times in the three workspaces, respectively.

VIII. Observation

RRT is a very simple algorithm to implement but it has its own cons. First, it is probabilistic complete due to the sampling method used. Hence, if the planner needs to be accurate it is not advisable to use RRT. The average lengths of the path in the workspaces are around [14, 17, 70] respectively. Whereas in the Updated Wavefront we always get the optimal path lengths [10, 15, 37]. Also, the proposed algorithm is slightly faster in computation than RRT.

IX. Conclusion

Subtopic 1: Implementing Wavefront planner and RRT planner on 2D workspaces.

Initially BNM was opted to find the initial feasible path. But this led to increase in computation time. Here, Wavefront planner proved to be faster to build. This subtopic was relatively easy to implement.

Subtopic 2: Turning using of Ackerman mechanism.

This was implemented in the Updated Wavefront Algorithm by calculating the radius of curvature of the path which was constructed using a two boundary value method. This subtopic needed a lot of polynomial calculations. We had 4 equations and a 4th order polynomial path segment could be constructed.

Subtopic 3: Implementing Path smoothing and Shooting method on RRT:

Initially AIT* and ABIT* were proposed for better results in sampling-based planners but with a kinematic planner with a greater number of independent control inputs, optimizing the nodes is difficult. [3] was used to implement this subtopic.

Subtopic 4: Implementing Control Inputs for the Ackerman Mechanism with both planners and compare them.

This subtopic was not completed. The closest this paper reached was plotting the paths and calculating the radius of curvature of the path for Updated Wavefront and the segments radii of segments of path in RRT. These values could be used in a simulation to visualize the actual motion of the car.

X. References

[1] Kamner, Haim J. "Combined articulated and ackerman steering system for vehicles." U.S. Patent No. 3,515,235. 2 Jun. 1970.

[2] LaValle, Steven M. "Rapidly-exploring random trees: A new tool for path planning." (199)

[3] Acharya, Rajaneesh, and Debashisha Jena. "Sampling based motion planning of Ackermann steering system using transformation." 2018 IEEMA Engineer Infinite Conference (eTechNxT). IEEE, 2018.

[4] Al-Jumaily, Adel, and Cindy Leung. "Wavefront propagation and fuzzy based autonomous navigation." International Journal of Advanced Robotic Systems 2.2 (2005): 10.

[5] https://datagenetics.com/blog/december12016/index.html

[6] Wahba, Grace. "Spline interpolation and smoothing on the sphere." SIAM Journal on Scientific and Statistical Computing 2.1 (1981): 5-16.

[7] https://en.wikipedia.org/wiki/Radius_of_curvature

[8] Strub, Marlin P., and Jonathan D. Gammell. "Adaptively Informed Trees (AIT*): Fast Asymptotically Optimal Path Planning through Adaptive Heuristics." arXiv preprint arXiv:2002.06599 (2020).

[9] Strub, Marlin P., and Jonathan D. Gammell. "Advanced BIT*(ABIT*): Sampling-Based Planning with Advanced Graph-Search Techniques." arXiv preprint arXiv:2002.06589 (2020).

[10] I. A. D. Nesnas, J. B. Matthews, P. Abad-Manterola, J. W. Burdick, J. A. Edlund, J. C. Morrison, R. D. Peters, M. M. Tanner, R. N. Miyake, B. S. Solish et al., “Axel and DuAxel rovers for the sustainable exploration of extreme terrains,” Journal of Field Robotics, vol. 29, no. 4, pp. 663–685, 2012

[11] Ha, Sung N. "A nonlinear shooting method for two-point boundary value problems." Computers & Mathematics with Applications 42.10-11 (2001): 1411-1420.