Path planning of a Non-holonomic car using Hybrid A* algorithm

Ajith Kumar Challa, Krishna Sai Avinash Pallela, Narahari Rahul Malayanur

Abstract:

The objective of this work is to develop and implement a path planning system for a non-holonomic car taking it from an initial configuration to a parking space. The parking spaces considered are parallel and perpendicular parking spaces and the planning algorithm considered is Hybrid A* algorithm. After the car reaches a point nearer to the parallel parking space and the orientation is as that of the goal, it implements a type-I maneuver for parallel parking.

Hybrid A* algorithm:

Since the cars have non-holonomic constraints, they cannot perform arbitrary rotations. In the context of this work, the car is modelled as follows: It can perform three actions which are:

(i) Go straight for a specified distance.
(ii) Left turn at a maximum steering angle.
(iii) Right turn at a maximum steering angle.

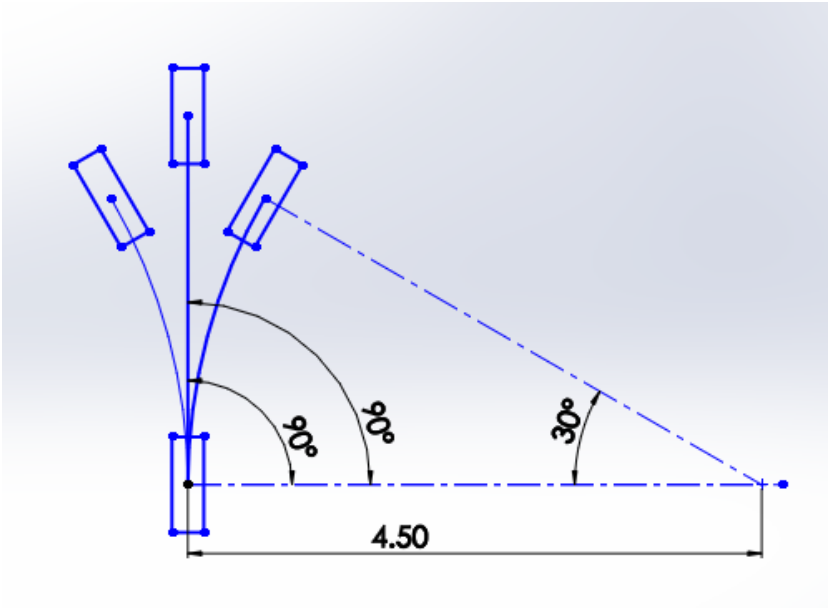

Figure 1:Straight, left and right children of a parent

The car's speed is not considered, and the inputs are the initial configuration of the car, $s_i = (x_i, y_i, \alpha_i)$ and the goal configuration, $s_g=(x_g, y_g, \alpha_g)$, where x and y are the location coordinates of the car in an obstacle map and α is the orientation of the car with respect to positive X-axis. The new children states are obtained from the kinematic model of the car using basic coordinate geometry. The heuristic (f = g + h) is considered, where h is the Euclidean distance (h) from the current configuration to the goal configuration and g is the accumulated cost from the initial configuration to the current configuration.

Kinematic model of the car:

The car's length and the minimum turning radius are considered such that they are proportionate, and in accordance with on-road vehicle dimensions. The dimensions considered are:

Length of the car = 2 units

Minimum turning radius of the car = 4.5 units

- **Calculating coordinates of the children:**

Generalized formulae using basic coordinate geometry are derived such that the left and right instantaneous centers of the car about which the car rotates is determined irrespective of the car's orientation. Then, the coordinates of the children are determined by rotating the car about the calculated instantaneous center and these points are verified by using a 3D Modeling software, Solidworks.

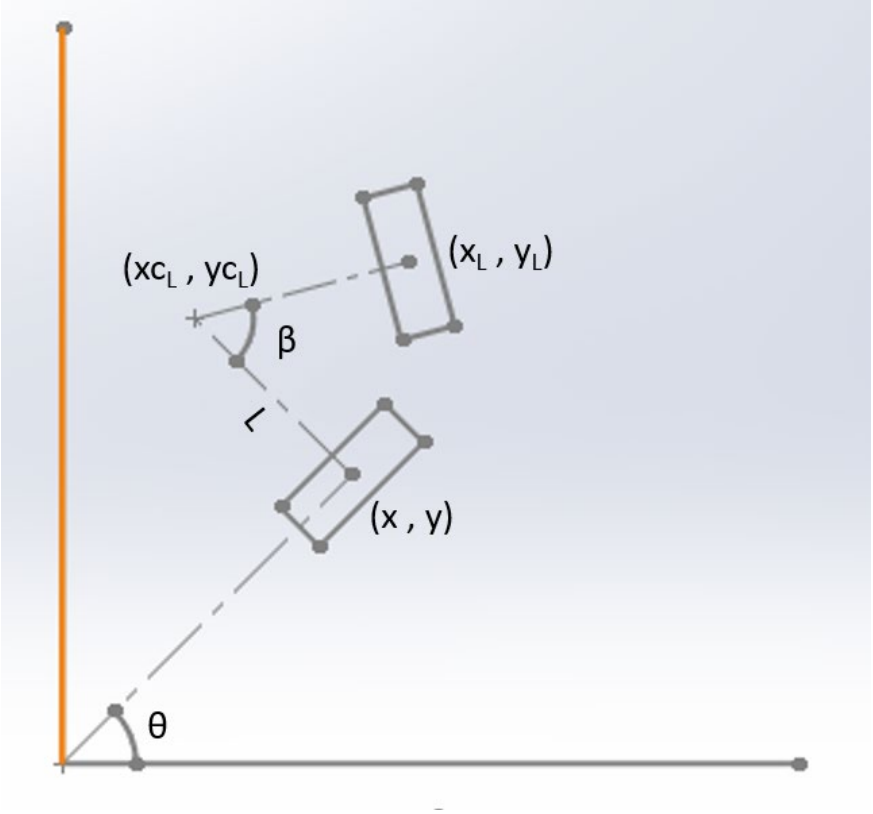

Figure 2:Geometrical representation of parent and child configurations

The above figure shows a geometrical representation of parent and child configurations

where, (x, y, θ) are the coordinates of the parent

(xc_L, yc_L) is the coordinate of the left instantaneous center

$(x_L, y_L, (\theta - \beta))$ is the coordinate of the left child

Deriving left child coordinates:

Considering two equations to calculate two unknowns xc_L and yc_L,

$$\tan(90 + \theta) = (yc_L - y) / (xc_L - x)$$

$$(xc_L - x)^2 + (yc_L - y)^2 = L^2$$

Using the above two equations, the coordinates of the left center are obtained as below,

$$yc_L = y + L*\cos\theta$$

$$xc_L = x + (y - yc_L)*\tan\theta$$

After calculating the centers, the coordinates of the children are calculated using the below two equations,

$$y_L = yc_L + L*\sin(\theta - \beta)$$

$$x_L = xc_L - [(yc_L - y) / \tan(\theta - \beta)]$$

Similarly, formulae for right instantaneous center and right child can be derived.

Implementation:

For our project, we have created various parking scenarios where the car must park parallelly either to its left or right parking space. We have also implemented a perpendicular parking scenario. Below is the pseudocode of our project:

Pseudocode for hybrid A*:

```
Initialize the start coordinate
OPEN.insert(start)  //  where OPEN is a priority queue
CLOSED = []
while !OPEN.empty()
  node = OPEN.pop()
  if node.STATE == GOAL return path (node)  ( For perpendicular parking )
  elif node.STATE < min distance of goal  ( For parallel parking )
    check if the parking space is to the left or to the right of the car
    go to nearest configuration of the parallel parking space and park parallelly
    start the type 1 maneuver to reach the goal
  else:
    calculate the three children
    if child.STATE not in OPEN or CLOSED
      OPEN.insert(child)
    else if child.STATE in OPEN with higher PATH-COST
       update that OPEN node with child
```

The hybrid A* algorithm is similar to the A* algorithm in which the nodes expanded are inserted in the open_set along with their cost. The least cost node is then popped from the inserted list and then it again checks the children of the given node. In our case, the parent has three children, one where car turns left with maximum steering angle, the other to its right and the third one is going straight from the current orientation.

By expanding these nodes based on the cost, the car tries to reach the goal position. In the case of a parallel parking scenario, the car initially reaches the nearest point and then starts type 1 maneuver.

Type-1 maneuver for parallel parking

The hybrid A* algorithm takes the car to a point less than a specified distance from the goal configuration and calculates ‘d’ and ‘delta theta’ to execute the type-I maneuver.

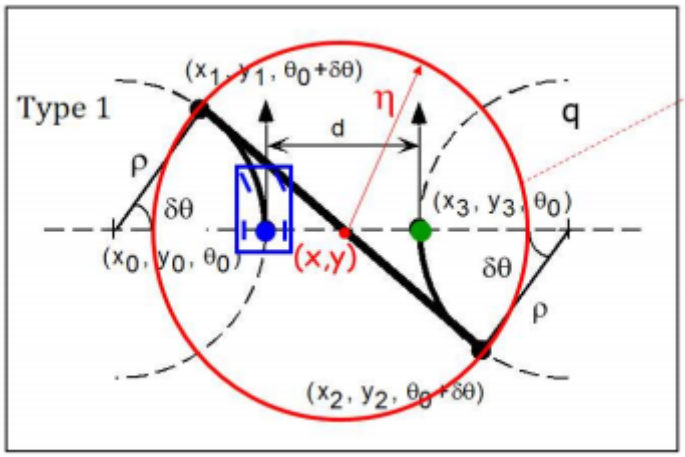

Figure 3:Type-I maneuver for parallel parking [2]

The algorithm is in such a way that the car tries to reach a point which is less than a specified distance from the goal position by having the same orientation as that of the goal configuration. After reaching the point, it calculates the distance of the goal position (in the figure3 mentioned as ‘d’) and then tries to calculate the steering angle required to reach the goal position (in the figure 3 mentioned as ‘delta theta’).

Results:

To show the robustness of the algorithm, multiple parallel parking scenarios are tested: one at the right side of the car and the other to its left and one at a different orientation. Once the car reaches a specific position, it checks which side is the parking space and based on that it either takes right or left which implements the parallel parking.

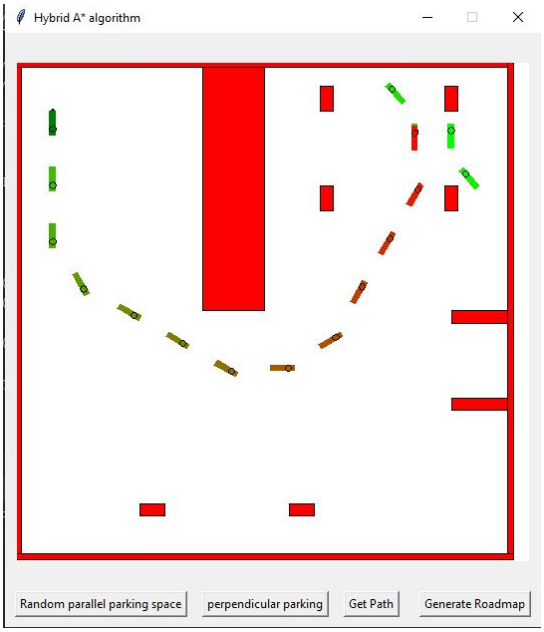

Figure 4:Parallel parking to the right side of the car

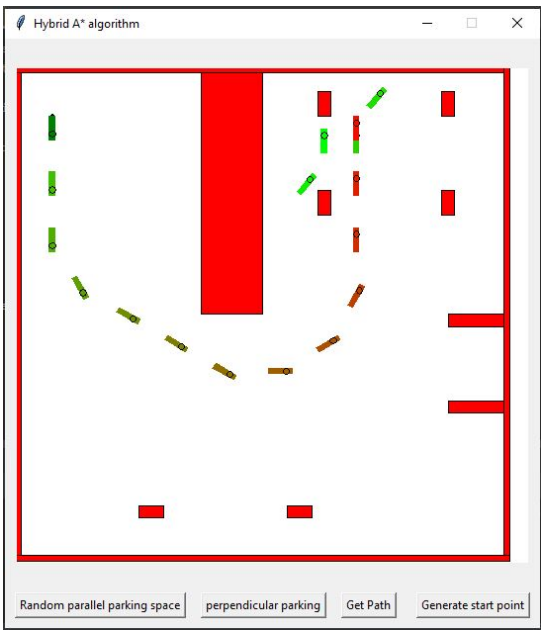

Figure 5: Parallel parking to the left side of the car

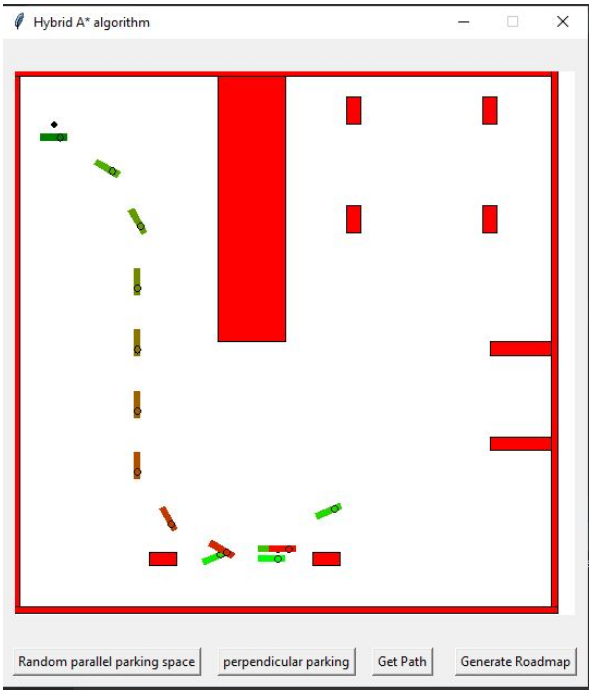

Figure 6:Parallel parking space at an altered orientation

In the above figure, the car reached to a lesser distance from the goal configuration than the previous iteration, but it still reached the goal by calculating the angle 'delta theta' by which it should rotate to reach the goal.

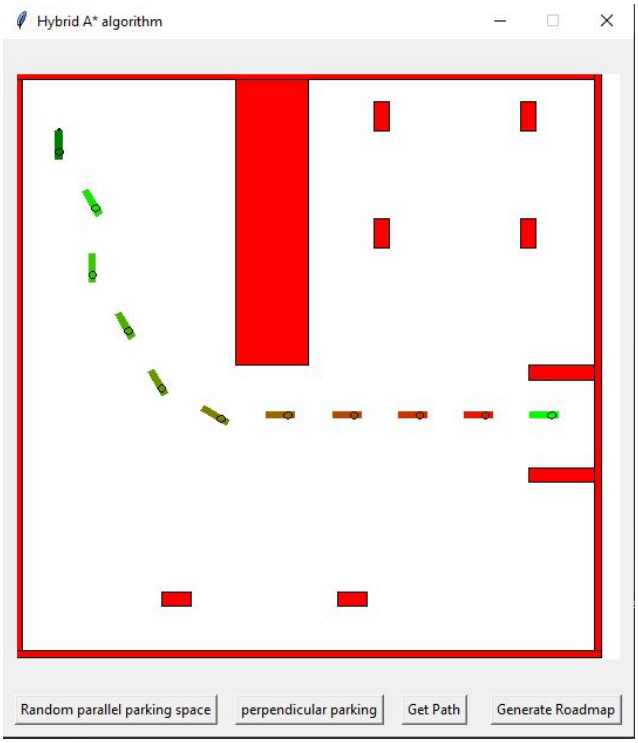

Figure 7: Perpendicular parking space

Observations and future works:

- **Heuristics** – In this project, heuristics are based on Euclidean distance between two points. Non-holonomic based obstacle heuristics can be used which would reduce the nodes expanded.
- **Kinematic model** – The problem can be accurately dealt by also considering the reverse motion of the car.
- **Varying steering inputs** – A car which has a fixed steering input has a very limited number of kinematically attainable configurations. This can be solved by increasing the number of steering angles and the steering of the on the road vehicles can be replicated by having a continuous steering input.

References:

[1] Dmitri, D. and Sebastian, T. (2010). Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments, *The International Journal of Robotics Research.*

[2] Ioannis Karamouzas, Assistant Professor at Clemson University, School of Computing.

[3] Class materials of CPSC 8810 – Special Topics on Motion Planning.