Autonomous Path Planning for a Car-Like Robot using Hybrid A* Search

Aashi Manglik (13006)
Supervisor: Prof. Gaurav Pandey

August 27, 2017

Abstract

Given the initial and goal 2D positions in the form of (x, y, θ), the aim is to find the shortest path which is drivable for a car-like robot considering its non-holonomic constraints. Non-holonomic constraints mean that the car cannot change its heading direction abruptly at a point. Thus, planning paths for a car is much more difficult than for holonomic agent like human. The hybrid A* search algorithm described in this report gives the optimal path as a sequence of continuous 2D coordinates along with the corresponding control.

1 Introduction

Hybrid A* search is a variant of A* search (applied to the 3D kinematic state space of the vehicle) to obtain a kinematically feasible trajectory [2]. There are two problems in using regular A* search algorithm for path planning of car-like robots. First, the world is continuous and A* states are discrete. Second, it does not consider the non-holonomic constraints of the vehicle. For example, a car cannot always execute a straight line motion between any two positions in the map. To overcome the first problem, hybrid A* assigns to each discrete cell in A* a continuous vehicle coordinate. This continuous coordinate is computed using the dynamic equations of motion of car given in section 2 and thus, this position is attainable by the actual robot. For mitigating second problem, the heuristic and cost of action is computed using dubins path described in 3.1. This dubins path length replaces the euclidean or manhattan distance used in conventional A*.

In hybrid A*, the vehicle state is represented in a 3-D discrete grid. Two of those dimensions represent the x-y-location of the vehicle center in smooth map coordinates and the third dimension is the vehicle's heading direction θ. Montemerlo et al [1] used a 4-D discrete grid where the fourth dimension indicates whether the vehicle is moving forward or in the reverse direction. The model of car used in this project assumes that the car can only move forward, i.e., it does not have a reverse gear.

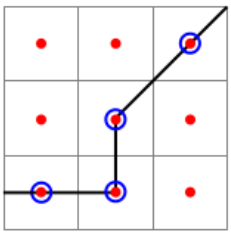

Figure 1: A* Path
Source: Montemerlo et al, Junior: The Stanford entry in the Urban Challenge

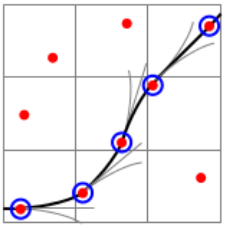

Figure 2: Hybrid-state A* Path
Source: Montemerlo et al, Junior: The Stanford entry in the Urban Challenge

2 Dynamic Equations of Car

(x, y, θ) - Initial 2D state
α - Steering angle
d - Distance by which rear tyre moved forward
β - Turning angle
R - Turning radius of car
$(x\prime, y\prime, \theta\prime)$ - Final 2D state of vehicle after applying the given α for distance d

Using Bicycle model for circular motion, the following equations are used to compute the final state. If $|\beta|$ is very small, then it is approximately a straight

line motion.

$$\beta = \frac{d}{L} \tan \alpha \tag{1}$$

$$If |\beta| < 0.001 : \tag{2}$$

$$x\prime = x + d \cdot \cos(\theta) \tag{3}$$

$$y\prime = y + d \cdot \sin(\theta) \tag{4}$$

$$\theta\prime = \theta + \beta \tag{5}$$

$$(\beta \approx 0 \implies \theta\prime \approx \theta) \tag{6}$$

$$else : \tag{7}$$

$$R = \frac{d}{\beta} \tag{8}$$

$$CX = x - (\sin \theta \cdot R) \tag{9}$$

$$CY = y - (\cos \theta \cdot R) \tag{10}$$

$$\theta\prime = (\theta + \beta) \mod 2\pi \tag{11}$$

$$x\prime = CX + \sin \theta\prime \cdot R \tag{12}$$

$$y\prime = CY - \cos \theta\prime \cdot R \tag{13}$$

3 Hybrid A* Search

3.1 Dubins Path

Dubin's car is a simplified model of car. It can only move forwards, never backwards and it is always moving with unit velocity thus not considering acceleration. Planning shortest path for dubins car requires simple geometric calculations whereas other dynamical systems require some high level and complex matrix operations.

The model assumes that the car essentially has only 3 controls - turn left at maximum steering angle denoted by L, turn right at maximum (R) and go straight (S). All paths traced by dubin's car are combinations of these three controls. Lester Dubins proved in paper [3] that there are only 6 combinations of these controls that describe all the shortest paths - LSR, RSL, LSL, RSR, LRL, RLR. LSR means that the car will first turn left from its initial position tracing the circle having minimum turning radius, then move along a tangent (straight line motion) to the circle which is at the right of goal position. The arc lengths traversed by the car and/or the length of straight line motion will vary depending on the initial and final position coordinates. In Figure 3, the green arrow q_s refers to the initial state of car and black arrow q_f to the desired goal orientation. Two circles of minimum turning radius, one at left and the other going right, are drawn from q_s and q_f each. Part (b) of Figure 3 represents RSL trajectory. In RSL, the car first turns right and traverses the arc upto certain length, then moves along the tangent between two circles and finally traverses an arc on the circle at the left of final position. The car attained the desired position by only moving forward. The geometric calculations for computing the best trajectory out of these 6 possibilities which include finding the centers of the circles, arc lengths, tangent points and finally the shortest path length is

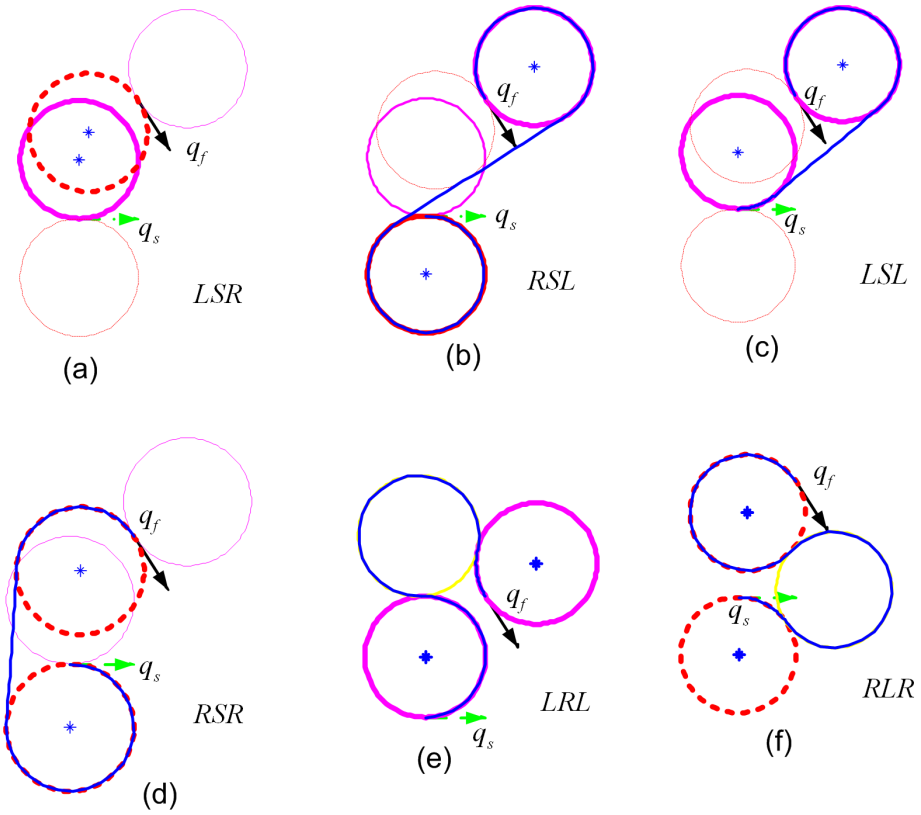

Figure 3: Dubins Curves
Source: https://i.stack.imgur.com/CFfQL.png

given in [4].

The minimum turning radius is provided by the user depending on the car's specifications.

3.2 Overview of A* search algorithm

In the standard terminology used, when talking about A*, $g(n)$ represents the exact cost of the path from the starting point to any vertex n, and $h(n)$ represents the heuristic estimated cost from vertex n to the goal. A* balances the two as it moves from the starting point to the goal. Each time through the main loop, it examines the vertex n that has the lowest $f(n)$ given by equation 14.

$$f(n) = g(n) + h(n) \tag{14}$$

3.3 Heuristics

Hybrid A* search is guided by two heuristics - nonholonomic without obstacles and holonomic with obstacles. Both of these heuristic values are estimated for each possible state in 3D discrete grid and maximum of the two is taken as the

final heuristic - $h(n)$. Greater the heuristic value, greater is the difficulty in reaching goal. Hence, taking the maximum makes sense as it takes into account the deciding factor - either external (obstacles in world map) or internal (car's dynamics) constraints.

3.3.1 Non-holonomic without obstacles

Assume a goal state of (x_g, y_g, θ_g) and compute the length of shortest Dubins path to the goal from every point (x, y, θ) in some discretized neighborhood of the goal, assuming complete absence of obstacles. This heuristic helps in pruning the search branches that approach goal with wrong headings. It can be fully precomputed offline since no run-time sensor information is required.

3.3.2 Holonomic with obstacles

It is the shortest distance to goal by performing dynamic programming in 2D. This heuristic value is computed by ignoring the heading direction of vehicle but using the obstacle map. If a shortest approach to goal is blocked by some obstacle, the cost of longer path is assigned to that cell. For example, Table 1 represents a world discretized into 3x4 grid where blue colored cells represent obstacles and the bottom right cell with value 0 indicates the goal. The value assigned to bottom left corner of grid would have been 3 in absence of obstacles but due to the presence of obstacles, it takes 7 steps instead of 3 to reach goal from that cell.

5	4	3	2
6		2	1
7		1	0

Table 1: Holonomic with obstacles heuristic computed using dynamic programming

3.4 Tree structure

The forward search in the algorithm uses a discretized space of control actions (steering). Let us denote the maximum steering angle of the car by α. The action space - ($-\alpha$, α) is discretized keeping a resolution of few degrees. Each continuous coordinate (x,y,θ) is associated with a cell in the grid and forms a node in the tree. All the possible actions in the discretised action space are applied to this node's coordinates and if the new position lies inside the grid, it becomes the child of that node. After getting all the children, the child whose $f(n)$ value is minimum is selected as the next position and the process repeats. This goes on until terminated by any of the following cases:

1. The child indicates the goal position. In this case, we have our desired optimal path.
2. The child has the minimum $f(n)$ value among its siblings but each action further takes it either outside the grid or will finally lead nowhere near the goal. In this case, we have to backtrace to the parent of that child and select different child based on the same $f(n)$ value criteria.

4 Experiments

4.1 Rviz Simulation

The heading direction of vehicle, denoted by θ is discretised in $(0,2\pi)$ keeping the resolution of 10°. For each cell in the grid, the x-y coordinates of center are chosen and non-holonomic heuristic is computed for each of the 36 possible values of θ. The cell size is taken to be 0.5 metres greater than the length of the car. For results given below the length of car is 2.5 metres, maximum steering angle is $\frac{\pi}{4}$ and the corresponding minimum turning radius is 1.76 metres. In each time step, the car travels a distance equal to its length. The following figures show the paths obtained from hybrid-state A* search algorithm and conventional A* algorithm in identical binary obstacle grids. The green cell indicates obstacle at that location.

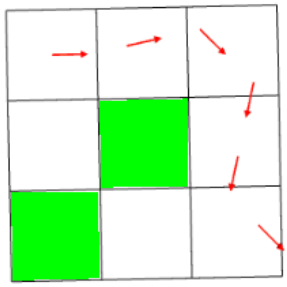

Figure 4: Initial Pose: (0,0,0), Final Pose: $(2,2,\frac{-\pi}{4})$ - Hybrid-state A* Path

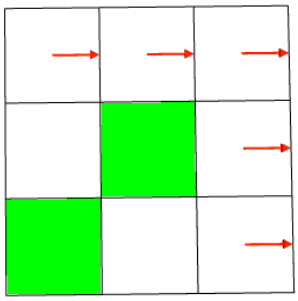

Figure 5: Initial Pose (0,0,0), Final Pose: $(2,2,\frac{-\pi}{4})$ - A* Path

4.2 Limitations of Hybrid-state A*

Though hybrid A* is guaranteed to yield drivable path, it may fail to find a path. The coarser the discretization, the more often hybrid A* will fail to find a path.

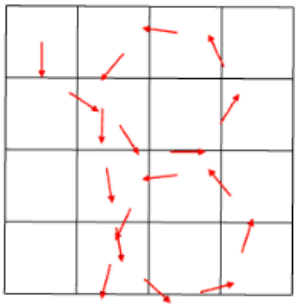

Figure 6: Initial Pose: $(0,0,-\frac{\pi}{2})$, Final Pose: $(3,1,-\frac{\pi}{2})$ - Hybrid-state A* Path

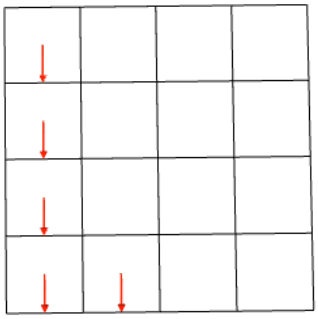

Figure 7: Initial Pose: $(0,0,-\frac{\pi}{2})$, Final Pose: $(3,1,-\frac{\pi}{2})$ - A* Path

While the paths are realizable by the vehicle, the small number of discrete actions available to the planner often leads to trajectories with rapid changes in steering angles which may require excessive steering. In [1], the path is further smoothed and optimized for minimum steering wheel motion.

5 Limitations of Dubins Model

Dubins proved that shortest path must be a smooth curve that is piecewise circular or linear, with at most 3 pieces, and always takes the form CCC or CSC where C is an arc of a circle and S is a line segment. To choose a path which returns to the initial point but in the opposite direction, two competing paths of Dubins type suggest themselves: $l_{\frac{3\pi}{2}} s_2 l_{\frac{3\pi}{2}}$ and $l_{\frac{\pi}{3}} r_{\frac{5\pi}{3}} l_{\frac{\pi}{3}}$ (see A, B in Figure 8). It can be seen that if the car has the ability to reverse, the paths are shorter (see C, D in Figure 8) and hence dubin's car model will not be applicable in this case. Reeds-Shepp [5] model of car is more appropriate in such cases. It is essentially the same thing as the Dubin's car, except it can also move backwards.

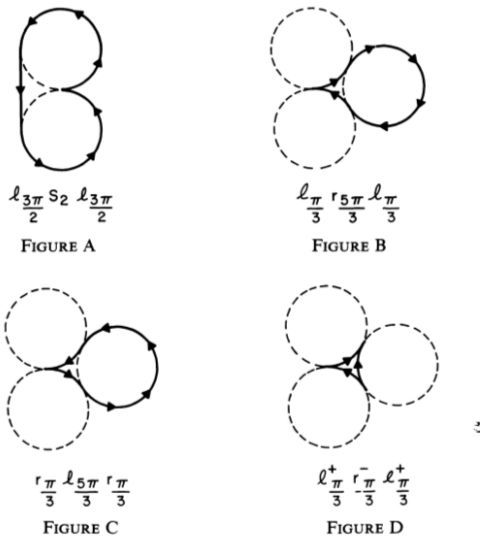

Figure 8: Dubins model's limitation [5]

6 Conclusion

A piecewise linear path found by plain A* cannot easily be executed; and even the much smoother Field D* [1] path possesses kinks that a vehicle cannot execute. By associating continuous coordinates with each grid cell in Hybrid A*, this approach results in a path that is executable in realistic environments. However, it is not guaranteed to find the minimal-cost solution because of the discretization of controls and time.

References

[1] Montemerlo et al, *Junior: The Stanford entry in the Urban Challenge.* Journal of Field Robotics - Special Issue on the 2007 DARPA Urban Challenge, Part II, September 2008

[2] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, James Diebel, *Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments.* The International Journal of Robotics Research, January-25-2010

[3] Dubins, L. E. (1957). *On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents.* American Journal of Mathematics.

[4] Andy Giese (2012), *A Comprehensive, Step-by-Step Tutorial on Computing Dubin's Curves.* Retrieved from https://gieseanw.files.wordpress.com/2012/10/dubins.pdf

[5] Reeds, J. A.; Shepp, L. A. Optimal paths for a car that goes both forwards and backwards. Pacific J. Math. 145 (1990), no. 2, 367–393