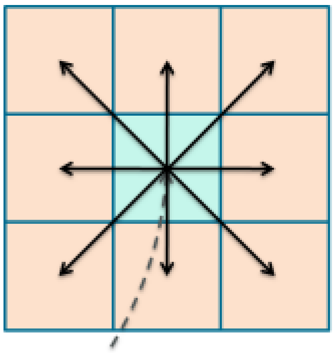
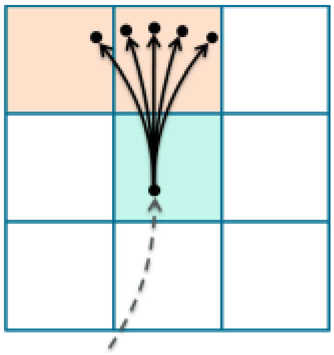
HYBRID A*

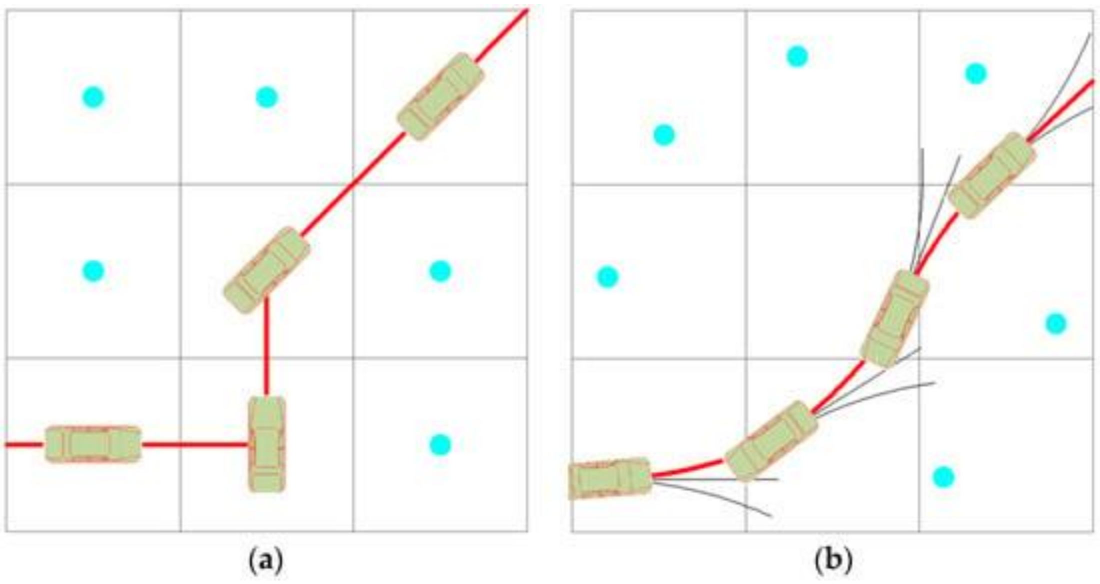
Imagine a cyclist travelling inside a forest filled with trees and bushes with which he could crash, if the cyclist thinks using the standard A* algorithm, he will perceive the bicycle as a vehicle with discrete movement that can turn instantaneously to avoid said trees or bushes, while if he thought using the Hybrid A* algorithm, he will take into account the continuous factor of the bicycles turning radius, and plan his path in consideration of this. In short, hybrid A* paths are a smooth curve of a standard A* path


(a) regular A*


(b) Hybrid A*

Standard A* maps the path as a cell where each state (or location) is only the centre of the cell. This produces a path of very sharp turns problematic for most vehicles. Hybrid A* takes these centre nodes and moves them inside the cell according to the turning radius of the object (or vehicle). This ensure that the path generated is traversable in real life applications, making it a far better alternative for the standard A* algorithm in automobile applications such as autopilot.

In standard A*, the path from one cell to another would be one straight line. However, in Hybrid A* more than one line is drawn representing the path from one cell to another. These lines are the different turns the vehicle can make (maximum right steering, maximum left steering, and forward). This is generally set to three lines emerging from a single node, and so, for every sub-path created exists a child node that traces back to the parent node (initial position of the vehicle).



For standard A*, the search space is represented by a 2D grid of x and y values. However, Hybrid A* utilizes another parameter θ which indicates the heading of the vehicle. This is the parameter that ensures the path constructed is feasible and can be implemented but makes the search space much larger than its standard counterpart.

You also need a few extra costs. According to the reports, you should add an extra cost to those nodes that change steering angle compared with the previous node, so you must save the steering angle in each node. You should also add extra costs if the node is close to an obstacle, or if you are reversing. The result is that you prioritize nodes that avoid obstacles because obstacles are bad and may scratch the vehicle. Additionally, you can use Euclidian distance as a heuristic as it is pretty much the most optimal estimate among the traditional heuristics. Although more advanced applications use a more complicated heuristic called **the Dubins or Reeds-Shepp path heuristic.** Frontier nodes also have different treatment when it comes to the Hybrid algorithm. The way you close nodes is that for each cell, you close it if the car has not been there before with a certain heading angle.