Path planning algorithms:

1] **RRT:**
The Rapidly–exploring Random Tree (RRT) algorithm is a popular technique for path planning with kinodynamic constraints. In simple terms, RRT builds a search tree of reachable states by attempting to apply random actions at known–reachable states. Unless the action causes the robot to make contact with an obstacle or violate some dynamics constraint, the action is considered successful, and the resulting state is added to the tree of reachable states.
The algorithm is as folllows:

- Randomly generate a point.
- If it falls on an obstacle remove it.
- Else extend it to the nearest node of the tree.
- Continue until a node is generated near the goal
- Retrace the path to the starting point.

Since RRT is a randomized algorithm, there is no guarantee that the goal will be found, even when a feasible path exists. In addition, due to the nature of exploration and extension used to generate the search tree, the final path may be jagged and meandering.

2]**RRT*:**
RRT has the advantage in providing the quickest first path and also probabilistic completeness but it does not ensure asymptotic optimality. In 2011, Karaman and Farazolli overcame this problem of asymptotic optimality by introducing the extended version of RRT known as RRT*. Rapidly Exploring Random Tree Star made it possible to reach an optimal solution but to do so, it has been proven to take an infinite time.

The algorithm is same as RRT. Apart from connecting the child to the parent node, RRT* also inspects each one of the nodes that are laid within the neighbourhood of the newly-added child. If one node could trace upto the tree root via child at a shorter distance than current connection, parent of this node shall be shifted to the child. The RRT* basically tries to smoothen tree branches. Mathematically it has been proved that with number of nodes reaching infinity, RRT* will be the shortest.

3] **PRM:**

- Mainly geared towards “multi-query” motion planning problems.
- Idea: build a graph (i.e., the roadmap) representing the “connectivity” of the environment; use this roadmap to figure out paths quickly at run time.

Algorithm:

- Sample n points in the free space
- Try to connect these points using a fast ‘local planner’ ignoring the obstacles
- If connection is successful, that is there is no collision, then add an edge.

During runtime,

- Connect the start and end goal to the closest nodes in the roadmap.
- Find a path on the roadmap.

PRM is probabilistically complete but NOT asymptotically optimal.

4] **RRG:**

Rapidly Exploring Random Graphs (RRGs), were designed to find feasible trajectories that satisfy specifications other than the usual “avoid all the obstacles and reach thegoal region”. More generally, RRGs can handle specifications given in the form of deterministic μ-calculus, which includes the widely-used Linear Temporal Logic (LTL). RRGs incrementally build a graph of trajectories, since specifications given in μ-calculus, in general, require infinite-horizon looping trajectories, which are not included in trees.
RRGs have similar algorithm as RRT. However, RRGs and RRTs differ in the choice of the vertices to be extended. Informally speaking, the RRT algorithm extends
the nearest vertex towards the sample. The RRG algorithm first extends the nearest vertex, and if such extension is successful, it also extends all the vertices returned by the Near procedure, producing a graph in general. In both cases, all the extensions resulting in collision-free trajectories are added to the graph as edges, and their terminal points as new vertices.