Path Planning Algorithms: Mathematical Formulations and Implementation

Adeel Ahsan
www.aeronautyy.com

September 4, 2025

1 Introduction

Path planning is a fundamental problem in robotics and autonomous systems, requiring the computation of collision-free paths from a start configuration to a goal configuration in the presence of obstacles. This report analyzes four distinct approaches to path planning, each with different theoretical foundations and practical characteristics.

2 Problem Formulation

Let $\mathcal{C} = [0, 1]^2$ represent the configuration space, and $\mathcal{C}_{obs} \subset \mathcal{C}$ represent the obstacle space consisting of triangular obstacles. The free space is defined as:

$$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$$

Given a start configuration $q_{start} \in \mathcal{C}_{free}$ and goal configuration $q_{goal} \in \mathcal{C}_{free}$, the path planning problem seeks to find a continuous path:

$$\pi : [0, 1] \to \mathcal{C}_{free}$$

such that $\pi(0) = q_{start}$ and $\pi(1) = q_{goal}$.

3 Rapidly-exploring Random Tree (RRT)

3.1 Mathematical Formulation

RRT constructs a tree $\mathcal{T} = (V, E)$ where V is the set of vertices (configurations) and E is the set of edges connecting vertices. The algorithm iteratively grows the tree as: The STEER function, which takes a current configuration and a target point and returns a new configuration a bounded step closer to the target, is defined as:

$$\text{STEER}(q_a, q_b, \Delta q) = \begin{cases} q_b & \text{if } \|q_b - q_a\| \leq \Delta q \\ q_a + \Delta q \cdot \frac{q_b - q_a}{\|q_b - q_a\|} & \text{otherwise} \end{cases}$$

3.2 Theoretical Properties

RRT is probabilistically complete, meaning that as $N_{max} \to \infty$, the probability of finding a solution (if one exists) approaches 1. However, RRT does not guarantee optimality.

Algorithm 1 RRT Algorithm

Initialize $\mathcal{T}$ with q_{start}
for $i = 1$ to N_{max} **do**
  Sample q_{rand} from $\mathcal{C}$ with probability $(1-p)$ or q_{goal} with probability p
  $q_{near} = \text{NEAREST}(\mathcal{T}, q_{rand})$
  $q_{new} = \text{STEER}(q_{near}, q_{rand}, \Delta q)$
  if COLLISION_FREE(q_{near}, q_{new}) **then**
    Add q_{new} to $\mathcal{T}$
    Add edge (q_{near}, q_{new}) to $\mathcal{T}$
    if $\|q_{new} - q_{goal}\| \leq \Delta q$ **then**
      return PATH(q_{start}, q_{goal})
    end if
  end if
end for

4 RRT* Algorithm

4.1 Mathematical Formulation

RRT* extends RRT by incorporating rewiring operations to improve path quality. The key addition is the concept of a neighborhood radius:

$$r_n = \min\left\{\gamma\left(\frac{\log n}{n}\right)^{1/d}, \eta\right\}$$

where n is the number of nodes, d is the dimension (2 in our case), $\gamma > 2^d(1+1/d)^{1/d}(\mu(\mathcal{C}_{free})/\zeta_d)^{1/d}$, and η is a maximum radius.

Algorithm 2 RRT* Key Operations

$\mathcal{X}_{near} = \text{NEAR}(\mathcal{T}, q_{new}, r_n)$
Choose parent $q_{min} = \arg\min_{q \in \mathcal{X}_{near}}\{\text{Cost}(q) + c(q, q_{new})\}$
for $q \in \mathcal{X}_{near}$ **do**
  if $\text{Cost}(q_{new}) + c(q_{new}, q) < \text{Cost}(q)$ **then**
    Rewire: set parent of q to q_{new}
  end if
end for

4.2 Theoretical Properties

RRT* is asymptotically optimal, meaning that as $n \to \infty$, the cost of the solution converges to the optimal cost c^*:

$$\lim_{n\to\infty} \mathbb{P}[\text{Cost}(\text{solution}) = c^*] = 1$$

5 Bidirectional RRT (BiRRT)

5.1 Mathematical Formulation

BiRRT maintains two trees: $\mathcal{T}_a$ rooted at q_{start} and $\mathcal{T}_b$ rooted at q_{goal}. The trees grow alternately toward each other:

Algorithm 3 BiRRT Algorithm

Initialize $\mathcal{T}_a$ with q_{start}, $\mathcal{T}_b$ with q_{goal}
for $i = 1$ to N_{max} **do**
  Sample q_{rand} from $\mathcal{C}$
  $q^a_{new} = \text{EXTEND}(\mathcal{T}_a, q_{rand})$
  if $q^a_{new} \neq \text{NULL}$ **then**
    $q^b_{new} = \text{EXTEND}(\mathcal{T}_b, q^a_{new})$
    if $\|q^a_{new} - q^b_{new}\| \leq \Delta q$ **then**
      return $\text{PATH}(q_{start}, q_{goal})$
    end if
  end if
  Swap $\mathcal{T}_a$ and $\mathcal{T}_b$
end for

5.2 Theoretical Properties

BiRRT typically finds solutions faster than single-tree RRT due to the bidirectional search, but maintains the same probabilistic completeness guarantee.

6 A* Search Algorithm

6.1 Mathematical Formulation

A* operates on a discretized grid representation of $\mathcal{C}$. Let $G = (V_G, E_G)$ be a graph where V_G represents grid cells and E_G represents valid transitions between adjacent cells. The algorithm maintains: - $g(n)$: actual cost from start to node n - $h(n)$: heuristic estimate from node n to goal - $f(n) = g(n) + h(n)$: estimated total cost

Algorithm 4 A* Algorithm

Initialize OPEN with start node, CLOSED $= \emptyset$
$g(\text{start}) = 0$, $f(\text{start}) = h(\text{start})$
while OPEN $\neq \emptyset$ **do**
  $n = \arg\min_{n' \in \text{OPEN}} f(n')$
  if $n =$ goal **then**
    return RECONSTRUCT_PATH(n)
  end if
  Move n from OPEN to CLOSED
  for each neighbor n' of n **do**
    if $n' \in$ CLOSED **then**
      continue
    end if
    $g_{tentative} = g(n) + c(n, n')$
    if $n' \notin$ OPEN OR $g_{tentative} < g(n')$ **then**
      $g(n') = g_{tentative}$
      $f(n') = g(n') + h(n')$
      Set parent of n' to n
      Add n' to OPEN if not already present
    end if
  end for
end while

6.2 Heuristic Function

The heuristic function used is the Euclidean distance:

$$h(n) = \sqrt{(x_{goal} - x_n)^2 + (y_{goal} - y_n)^2}$$

This heuristic is admissible (never overestimates the true cost) and consistent.

6.3 Improvements for Oscillation Reduction

To address oscillatory behavior, several improvements were implemented:

6.3.1 8-Connected Grid

Instead of 4-connected movement, 8-connected movement is used with appropriate cost weighting:

$$c(n, n') = \begin{cases} 1 & \text{if orthogonal movement} \\ \sqrt{2} & \text{if diagonal movement} \end{cases}$$

6.3.2 Path Smoothing

A post-processing step removes unnecessary waypoints using line-of-sight checks:

Algorithm 5 Path Smoothing

```
smoothed = [p_0]
i = 0
while i < |path| - 1 do
   farthest = i + 1
   for j = i + 2 to |path| - 1 do
      if COLLISION_FREE(p_i, p_j) then
         farthest = j
      else
         break
      end if
   end for
   Add p_farthest to smoothed
   i = farthest
end while
```

7 Implementation Details

7.1 Obstacle Representation

Obstacles are represented as triangular polygons generated using Delaunay triangulation with alpha-shape filtering:

$$R = \frac{abc}{4A}$$

where a, b, c are side lengths and A is the triangle area. Triangles with $R < 1/\alpha$ are kept as obstacles.

7.2 Collision Detection

Collision detection uses geometric intersection tests such as point-in-triangle using barycentric coordinates and segment-segment intersection using orientation tests.

8 Conclusion

This report presented mathematical formulations and implementations of four path planning algorithms. Each algorithm offers different trade-offs between computational efficiency, solution quality, and implementation complexity.

References

[1] LaValle, S. M. (2006). *Planning algorithms.* Cambridge University Press.

[2] Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7), 846-894.

[3] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100-107.