ECE 276B Project 2 Report

Qihao Qian

UCSD ECE Department
A69030355
q2qian@ucsd.edu

I. Introduction

Motion planning in complex three-dimensional environments is critical for various applications such as robotics, autonomous navigation, and virtual reality simulations. The primary objective of this project is to implement and analyze two prominent motion planning algorithms, namely the Weighted A* algorithm and the Bi-directional Rapidly-exploring Random Tree (Bi-RRT) algorithm, to effectively navigate a robot from a specified start position to a goal position within a given obstacle-filled workspace.

The Weighted A* algorithm extends the classic A* search by incorporating a heuristic weight factor, thereby controlling the balance between path optimality and computational efficiency. Specifically, it utilizes a discretized grid structure to systematically explore and prioritize paths based on their heuristic estimates, ensuring optimal or near-optimal solutions within bounded computation time.

On the other hand, the Bi-directional Rapidly-exploring Random Tree (Bi-RRT) algorithm employs a probabilistic sampling approach to efficiently explore the continuous search space. This method constructs two trees simultaneously, rooted respectively at the start and goal configurations. The trees iteratively expand by randomly sampling configurations, steering toward these samples, and incrementally building collision-free paths. Upon the intersection of these two trees, a viable path connecting the start and goal positions is established. This approach significantly accelerates path discovery, especially in high-dimensional or densely cluttered environments.

To rigorously assess these algorithms' performance, several benchmark scenarios are evaluated, encompassing varying complexities and obstacle distributions. Key performance metrics considered include computational time and path length, thereby highlighting each method's strengths and limitations. By comparing Weighted A* and Bi-RRT across diverse test cases, this study aims to provide insights into their practical applicability, informing algorithm selection for real-world robotics and navigation challenges.

II. Problem Statement

Consider a deterministic motion planning problem within a continuous three-dimensional Euclidean space $\mathbb{R}^3$. The workspace is defined by an axis-aligned rectangular boundary:

$$\mathcal{W} = \big\{(x,y,z) \;\big|\; \begin{array}{c} x_{\min} \le x \le x_{\max}, \\ y_{\min} \le y \le y_{\max}, \\ z_{\min} \le z \le z_{\max} \end{array} \big\}.$$

Within this workspace, there exists a finite set of static obstacles, represented by Axis-Aligned Bounding Boxes (AABBs), where each obstacle is defined as:

$$\mathcal{O}_i = \big\{(x,y,z) \;\big|\; \begin{array}{c} x_{i,\min} \le x \le x_{i,\max}, \\ y_{i,\min} \le y \le y_{i,\max}, \\ z_{i,\min} \le z \le z_{i,\max} \end{array} \big\}, \quad i = 1,\dots,N.$$

The free space $\mathcal{F}$ is thus the region available for the robot to navigate, defined by subtracting obstacles from the workspace:

$$\mathcal{F} = \mathcal{W} \setminus \bigcup_{i=1}^{N} \mathcal{O}_i.$$

Given an initial position $x_s \in \mathcal{F}$ (start) and a goal position $x_g \in \mathcal{F}$, the fundamental objective is to find a collision-free path connecting these two points. Formally, such a path can be mathematically described as a continuous and differentiable curve parameterized by $t \in [0,1]$:

$$\sigma : [0,1] \to \mathcal{F}, \quad \begin{cases} \sigma(0) = x_s, \\ \sigma(1) = x_g. \end{cases}$$

The deterministic shortest path problem explicitly seeks to minimize the length of the path. The length of a path is defined by the integral of the Euclidean norm of its derivative, representing the total traveled distance in Euclidean space:

$$\text{minimize} \quad L(\sigma) = \int_0^1 \|\dot{\sigma}(t)\|\, dt,$$

subject to the constraint that the path must remain collision-free:

$$\sigma(t) \in \mathcal{F}, \quad \forall t \in [0,1].$$

Practically, ensuring collision-free conditions involves rigorous checking of the path against obstacles, typically by using computational geometry techniques such as segment-to-AABB intersection tests. The provided methods incorporate these checks explicitly, iterating over potential configurations to guarantee obstacle avoidance.

To address the complexity of finding such a path, two different approaches are utilized: the Weighted A* algorithm

and the Bi-directional Rapidly-exploring Random Tree (Bi-RRT) algorithm. The Weighted A* algorithm discretizes the continuous space into a structured grid and systematically explores feasible paths using a weighted heuristic function to guide the search efficiently towards the goal. The Bi-RRT algorithm, on the other hand, performs a randomized, probabilistic search by simultaneously extending two trees from the start and goal configurations, significantly enhancing search efficiency in high-dimensional and obstacle-rich environments.

Ultimately, these algorithms aim not only to find feasible solutions but to balance computational complexity with the quality and optimality of the discovered paths.

III. Technical Approach

A. Collision Checking (Part 1)

To ensure safe planning in continuous 3D space, we implement exact intersection tests between the line segment $\overline{\mathbf{p}\_0\mathbf{p}\_1}$ and axis-aligned bounding boxes (AABBs) using the slab method. For an AABB defined by

$$\text{AABB} = [x_{\min}, y_{\min}, z_{\min}, x_{\max}, y_{\max}, z_{\max}]$$

and segment endpoints $\mathbf{p}\_0, \mathbf{p}\_1 \in \mathbb{R}^3$, we parametrize the line:

$$\mathbf{p}(t) = \mathbf{p}_0 + t\,(\mathbf{p}_1 - \mathbf{p}_0), \quad t \in [0,1].$$

For each axis $i \in x, y, z$, define

$$d_i = (\mathbf{p}_1 - \mathbf{p}_0)_i.$$

If $|d\_i| < \epsilon$, the segment is parallel to the i-slab; we reject immediately if

$$p_{0,i} \notin [x_{\min},\, x_{\max}].$$

Otherwise compute entry and exit parameters:

$$t_{i,\text{near}} = \frac{x_{\min,i} - p_{0,i}}{d_i},$$
$$t_{i,\text{far}} = \frac{x_{\max,i} - p_{0,i}}{d_i}.$$

Update the global intersection interval:

$$t_{\min} = \max_i\big(\min\{t_{i,\text{near}}, t_{i,\text{far}}\}\big),$$
$$t_{\max} = \min_i\big(\max\{t_{i,\text{near}}, t_{i,\text{far}}\}\big).$$

A collision occurs if and only if

$$t_{\min} \le t_{\max} \quad \text{and} \quad [t_{\min}, t_{\max}] \cap [0,1] \neq \emptyset.$$

We encapsulate this in a fast vectorized routine, first verifying both segment endpoints lie inside the global boundary before testing obstacles.

B. Weighted A* Planner(Part 2)

Given start $\mathbf{s}$ and goal $\mathbf{g}$ in $\mathbb{R}^3$, we discretize space on a grid of resolution r and use weight $w \ge 1$.

1. State Discretization.

$$\mathbf{k} = \text{key}(\mathbf{p}) = \left\lfloor \frac{\mathbf{p}}{r} \right\rceil \in \mathbb{Z}^3,$$

where $\lfloor\cdot\rceil$ rounds each component.

2. Neighbor Directions.

$$\mathcal{D}_0 = \{\,(d_x,d_y,d_z)\,|\,d_i \in \{-1,0,1\},\ (d_x,d_y,d_z) \neq (0,0,0)\},$$
$$\mathcal{D} = \big\{\, r\,\mathbf{d}/\|\mathbf{d}\|_2 : \mathbf{d} \in \mathcal{D}_0 \big\}.$$

3. Costs.

$$g(\mathbf{k}) : \text{cost-to-come},$$
$$h(\mathbf{k}) = \big\|\, r\,\mathbf{k} - \mathbf{g}\big\|_2,$$
$$f(\mathbf{k}) = g(\mathbf{k}) \ + \ w\,h(\mathbf{k}).$$

4. Collision Checking. Test each edge $\mathbf{p} \to \mathbf{n}$ with `is_segment_collision_free`$(\mathbf{p}, \mathbf{n})$.

5. Search Loop.

1) Initialize heap with $\big(f(\text{key}(\mathbf{s})), \text{key}(\mathbf{s})\big)$.
2) While heap not empty:
   - Pop $\mathbf{k}_{\text{cur}}$ with smallest f.
   - If $\mathbf{k}_{\text{cur}} = \text{key}(\mathbf{g})$, stop.
   - Else for each $\delta \in \mathcal{D}$:

$$\mathbf{n} = r\,\mathbf{k}_{\text{cur}} + \delta,$$
$$\mathbf{k}_{\text{nbr}} = \text{key}(\mathbf{n}).$$

If collision-free and $g(\mathbf{k}_{\text{cur}}) + \|\delta\|_2 < g(\mathbf{k}_{\text{nbr}})$, update

$$g(\mathbf{k}_{\text{nbr}}) = g(\mathbf{k}_{\text{cur}}) + \|\delta\|_2, \quad f(\mathbf{k}_{\text{nbr}}) = g(\mathbf{k}_{\text{nbr}}) + w\,h(\mathbf{k}_{\text{nbr}}),$$

push into heap and record predecessor.

6. Path Reconstruction. Backtrack from $\text{key}(\mathbf{g})$ through predecessors, then map keys $\mathbf{k}_i$ back to $\mathbf{p}_i = r\,\mathbf{k}_i$.

Properties:

- *Suboptimality:* $c \le w\,c^*$.
- *Completeness:* Resolution-complete at grid r.
- *Time:* $O(N \log N)$ heap ops.
- *Memory:* $O(N)$ states & heap.

C. Bi-directional RRT Planner(Part 3)

The Bi-directional RRT grows two trees $\mathcal{T}_a$ (from $\mathbf{s}$) and $\mathcal{T}_b$ (from $\mathbf{g}$) and attempts to connect them.

1. Initialization. $\mathcal{T}_a.V = \{\mathbf{s}\}$, $\mathcal{T}_a.P = \{-1\}$, $\mathcal{T}_b.V = \{\mathbf{g}\}$, $\mathcal{T}_b.P = \{-1\}$.

2. Sampling. With probability p_{goal}, sample the root of the other tree; otherwise sample uniformly:

$$q_{\text{rand}} = \begin{cases} \text{root}(\mathcal{T}_b), & \text{w.p. } p_{\text{goal}}, \\ \text{Uniform(boundary)}, & \text{else.} \end{cases}$$

3. Nearest and Steer. Find $i^* = \arg\min_i \|\mathcal{T}_a.V_i - q_{\text{rand}}\|$, let $q_{\text{near}} = \mathcal{T}_a.V_{i^*}$. Then

$$q_{\text{new}} = q_{\text{near}} + \frac{\min(\Delta, q_{\text{rand}} - q_{\text{near }2})}{\|q_{\text{rand}} - q_{\text{near}}\|_2}(q_{\text{rand}} - q_{\text{near}}),$$

where Δ is the step size.

4. Extend. If q_{new} is inside bounds and collision-free, append to $\mathcal{T}_a$: $\mathcal{T}_a.V \leftarrow \mathcal{T}_a.V \cup \{q_{\text{new}}\}$, record its parent index.

5. Connect Opposite Tree. Let $q_{\text{target}} = q_{\text{new}}$. Iteratively steer from the nearest vertex in $\mathcal{T}_b$ toward q_{target}, adding collision-free segments until $\|q_{\text{cur}} - q_{\text{target}}\|_2 < \frac{\Delta}{2}$ or blocked. If connected, record index j_{conn}.

6. Path Extraction. If trees connect, backtrack from q_{new} in $\mathcal{T}_a$ to $\mathbf{s}$ and from j_{conn} in $\mathcal{T}_b$ to $\mathbf{g}$, then concatenate.

Properties:

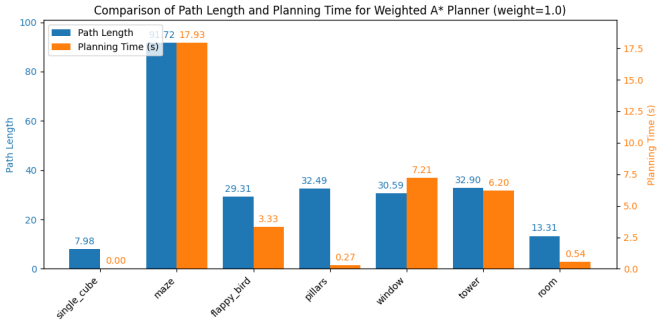
- *Probabilistic Completeness:* As $\max\_\text{iter} \to \infty$, success probability $\to 1$.
- *Expected Time:* $O(n)$ per iteration for nearest-neighbor (can be accelerated).
- *Quality:* No optimality guarantee, but practical with small Δ.

IV. Result

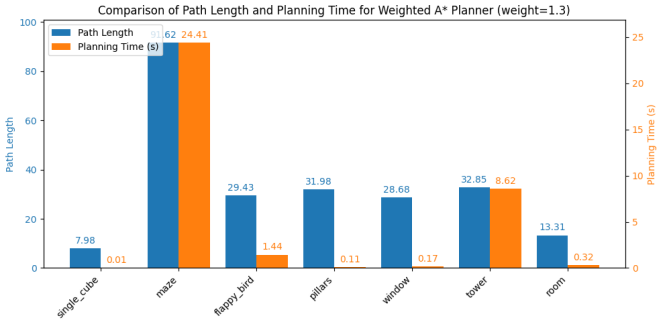
Tested on every map given by the project, and compare speed and path length of 2 algorithm.

A. Weighted A* Planner

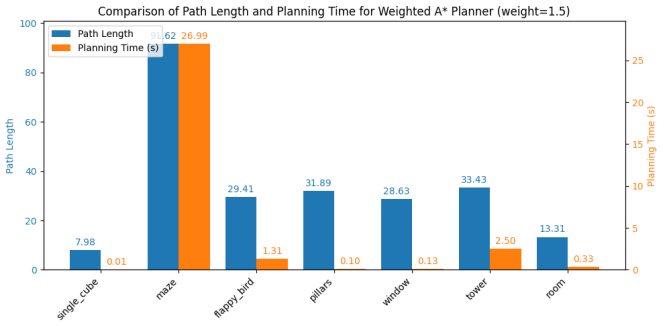
For different weight, performance is different.



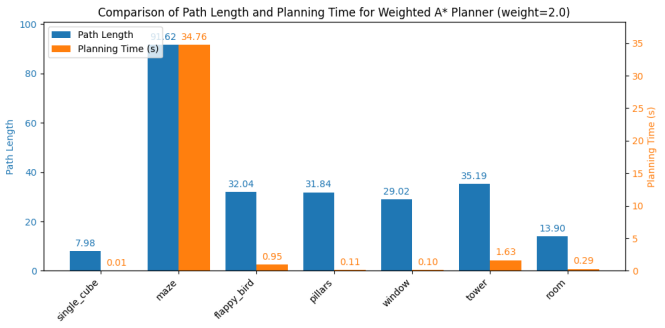
(a) weight=1



(b) weight=1.3



(c) weight=1.5



(d) weight=2

Fig. 1: comparision of Weighted A* planner for different weight

B. Bi-Directional RRT Planner

Much faster than Weighted A* Planner, but not the best path.

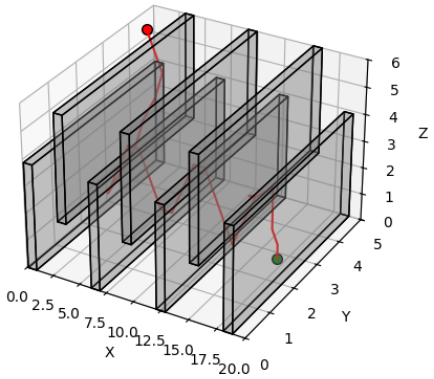


Fig. 2: Weighted A* Planner for flappy bird Map

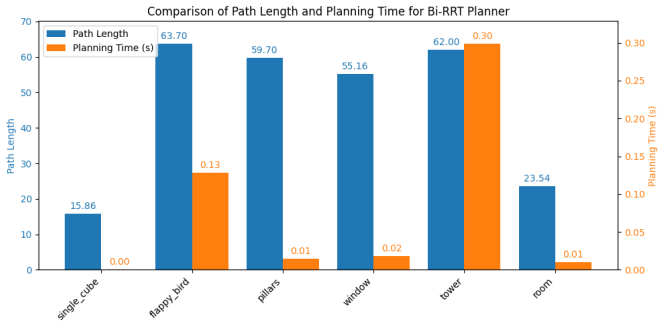


Fig. 3: Bidirectional RRT Planner Performance

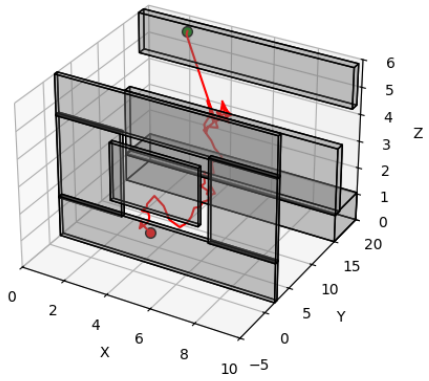


Fig. 4: Bidirectional RRT Planner for window