# Project 2: Motion Planning

1st Aditya Mishra
*Department of Mechanical and Aerospace Engineering*
*University of California San Diego*

## I. Introduction

This literature presents a comparison between search based algorithm and sampling based algorithm to find an efficient path between two points in a known three-dimensionally restricted environment with obstacles. Here the definition of efficient being the shortest possible path which can be calculated in shortest possible time. The optimization stresses on shortness of the path between the two points and the time taken to evaluate the path. There are seven different known three-dimensional environments where these algorithms are tested.
In this literature weighted A* algorithm is tested for search based algorithm. This algorithm does not guarantee an optimal path between the two given points if the weighting is not equal to one, but it is considerably faster to compute a feasible path to the goal.
For sampling based algorithm RRT* algorithm is tested for most environments. For the Monza environment, bi-directional RRT* algorithm is used since it produces the result quicker.
Each given environment is a restricted environment with boundaries and also has box-shaped obstacles of unique dimensions and have unique arrangement. This literature also talks about the collision detection algorithm developed for finding a collision free path. This algorithm is significantly used to detect if the returned path from RRT* or bi-directional RRT* algorithms are obstacle free since the library used returns unreliable results for these algorithms. ***Index Terms*—Weighted A*, RRT*, Bi-Directional RRT***

## II. Problem Statement

### A. State Space

The state $x$ is a vector consisting the location of the agent in terms of the u,v and w coordinates which specify the location of the agent with respect to the origin $(0, 0, 0)$.

$$x = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \text{ where, } u, v, w \in \mathbb{R} \tag{1}$$

Since the testing environments are restricted environments, each coordinate of $x$ is constrained to upper and lower bounds given by:

$$\begin{aligned} u_{min} \leq u \leq u_{max} \\ v_{min} \leq v \leq v_{max} \\ w_{min} \leq w \leq w_{max} \end{aligned} \tag{2}$$

Let the vector space $\mathbb{E}$ denote a space $\forall x$ which are subjected to the constraint mentioned in (2). Let $\mathbb{B}_i$ be a vector space of all $x$ satisfying 2 such that if $x \in \mathbb{B}$ then

$$\begin{aligned} u_{min}^{B_i} \leq u \leq u_{max}^{B_i} \\ v_{min}^{B_i} \leq v \leq v_{max}^{B_i} \\ w_{min}^{B_i} \leq w \leq w_{max}^{B_i} \\ \forall u, v, w \in \mathbb{B}_i \end{aligned} \tag{3}$$

Here $u_{min}^{B_i}$ and $u_{max}^{B_i}$ are the $u$ coordinates of the extreme corners of the obstacle block $B_i$. The same applies for $v$ and $w$ coordinates. Let $\mathbb{O}$ set of $\mathbb{B}_i \forall i$ present in the environment such that if $x \in \mathbb{O} \implies x \in \mathbb{B}_i$ for any $i$.
Hence the control space $\chi$ can be defined as the following

$$x \in \chi \implies x \in \mathbb{E} \text{ and } x \notin \mathbb{O} \tag{4}$$

### B. Control Space

The control space $\mathcal{U}$ is the set of given actions taken by the agent to reach the final goal $(\tau)$. $|\mathcal{U}| = 26$ since in this three dimensional space there are possibly 26 states neighboring any current state of the agent. The control is $u$ is taken to reach these neighboring states. In the search based algorithm a discrete magnitude $d$ is chosen as the resolution i.e. the agent at one instant can take $d$ or $-d$ step in any or every coordinate $u$, $v$ and $w$.

$$u \in \{(\pm d, 0, 0), (0, \pm d, 0), (0, 0, \pm d), (\pm d, \pm d, 0), ...., (\pm d, \pm d, \pm d)\} \tag{5}$$

$$\forall u \in \mathcal{U}$$

### C. Motion Model

Based on the definition of the state space given in (4) and the constraints given in (2) it is pretty evident that the control $u$ chosen from the control space given in (5) will abide by the constraints imposed by the restricted environment and the obstacles. The control $u$ can be called a valid control if

$$\begin{aligned} x_{t+1} = x_t + u_t \\ x_{t+1} \in \mathbb{E} \text{ and } \notin \mathbb{O} \\ \forall x_t \in \mathbb{E} \text{ and } \notin \mathbb{O} \end{aligned} \tag{6}$$

### D. Planning Horizon

The algorithm ends when the distance of the final point on the path from $(\tau)$ is less than 0.1.
$||x_{final} - x_{goal}|| <= 0.1$
Hence let the space $\mathbb{F}$ denote all such vectors. Then the planning horizon can be formulated as: $T = |\mathbb{E}| - |\mathbb{O}| - |\mathbb{F}|$

### E. Heuristic, Terminal cost and stage cost

for stage cost $l$, terminal cost $q$, and heuristic $h$

$$l(x_t, u_t) = |u_t|$$
$$q(x_t) = 0$$
$$h(x_t) = ||x_{goal} - x_t||$$
$$\forall x_t, u_t$$

## III. Technical Approach

### A. Search Based Planning algorithm

Initially the restricted environment is discretized with a resolution of 0.2 in every coordinate direction. Then the coordinates which lie inside the obstacles are assigned $-10$ to the value of $g$. The initial position of the agent is given $g$ value as 0. Rest of the points are given an arbitrarily high value for $g$, on this case $\infty$. After each g values are assigned, weighted A* algorithm is used to find the most feasible path from start to goal.

Weighted A* Algorithm

$OPEN \leftarrow \{s\}, CLOSED \leftarrow \{\}, \epsilon \geq 1$
$g_s = 0, g_i = \infty \ \forall i \in \mathbb{O}$
$g_i = 10000 \forall$ i $\in \mathbb{E} - \mathbb{O} - \{s\}$
**while** $\tau \notin$ CLOSED **do**
  Remove $i$ with smallest $f_i = g_i + \epsilon h_i$
  Insert $i$ into CLOSED
  **for** $j \in$ Children$(i)$ and $j \notin$ CLOSED **do**
    **if** $g_i + c_{ij} < g_j$ and $g_j > 0$ **then**
      $g_j \leftarrow g_i + c_{ij}$
      $Parent(j) \leftarrow i$
      **if** $j \in$ OPEN **then**
        Update priority of $j$
      **else**
        OPEN $\leftarrow$ OPEN $\bigcup\{j\}$
      **end if**
    **end if**
  **end for**
**end while**

*1) Optimality:* Since this algorithm uses $\epsilon$-consistent heuristic, it guarantees to return an $\epsilon$-suboptimal path with cost dist(s,$\tau$)$\leq g_\tau \leq \epsilon$ dist(s,$\tau$) for $\epsilon \geq 1$. In this literature the algorithm is iterated on various $\epsilon$, and the value of 3 is chosen for most of the environments.

*2) Completeness:* Sine the environment is discretized into a finite set of points and all the edge cots are non-negative, the weighted A* algorithm is guaranteed to return a path if it exists and hence it is complete.

*3) Memory:* Let us assume that each coordinate is discretized to to n points, then the memory complexity of this algorithm is $\mathcal{O}(n^3)$.

*4) Time efficiency:* The implementation can be seen in two steps. The first one being the discretization of the 3D environment and assigning $g$ values to every coordinate. The time complexity for this is $\mathcal{O}(n^3)$. The second part being the weighted A star algorithm. In the code priority queue was implemented using pqdict(). If the total number of points in the space where $|V|$ then the time complexity of this part ids $\mathcal{O}(|V|^2)$.

### B. Sampling based algorithms

In this case two algorithms are used which are RRT* and Bi-directional RRT*. The only difference in the implementation of the RRT* to that from the slides is that at every iteration there is probability to check a direct connection to the goal from the recent node explored. RRT* Algorithm

$V \leftarrow x_s$, $E \leftarrow \phi$
**for** i=1.....n **do**
  $x_{rand} \leftarrow SAMPLEFREE()$
  $x_{nearest} \leftarrow NEAREST((V, E), x_{rand}$
  $x_{new} \leftarrow STEER(x_{nearest}, x_{rand})$
  **if** $COLLISIONFREE(x_{nearest}, x_{new})$ **then**
    $X_{near} \leftarrow NEAR((V, E), x_{new}, min\{r*, \epsilon\})$
    $V \leftarrow V \bigcup\{x_{new}\}$
    $c_{min} \leftarrow COST(x_{nearest}) + COST(Line(x_{nearest}, x_{new}))$
    **for** $x_{near} \in X_{near}$ **do**
      **if** $COLLISIONFREE(x_{near}, x_{new})$ **then**
        **if** $COST(x_{near}) + COST(Line(x_{near}, x_{new})) < c_{min}$ **then**
          $x_{min} \leftarrow x_{new}$
          $c_{min} \leftarrow COST(x_{near}) + COST(Line(x_{near}, x_{new}))$
        **end if**
      **end if**
    **end for**
    $E \leftarrow E \bigcup\{(x_{min}, x_{new})\}$
    **if** $RANDOM(pr) \geq 0.5$ **then**
      **if** $COLLISIONFREE(x_{new}, x_\tau)$ **then**
        $E \leftarrow E \bigcup\{(x_{new}, x_{tau})\}$
        **return** G=(V,E)
      **end if**
    **end if**
    **for** $x_{near} \in X_{near}$ **do**
      **if** $COLLISIONFREE(x_{new}, x_{near})$ **then**
        **if** $COST(x_{near}) + COST(Line(x_{near}, x_{new})) < COST(x_{near})$ **then**
          $x_{parent} \leftarrow PARENT(x_{near})$
          $E \leftarrow (E \{(x_{parent}, x_{near})\}) \bigcup\{(x_{new}, x_{near})\}$
        **end if**
      **end if**
    **end for**
  **end if**
**end for**
**return** G=(V,E)

*1) Parameters used by the library:*

- rewire_count: number of nearby neighbors. Value assigned =26
- max_sample: max number of iterations (n) = 2,000,000

- r: discretization of edges to check for collision = 0.005. The lower the value, the longer it will take for the code to run.
- prc: probability of checking direct connection from the node at every iteration: 0.01
- Q: list of all the edge lengths and total number of edges corresponding to the lengths= Q=[(0.1,6),(0.1*2**0.5,18),(0.1*3**0.5,2)]

The collision detection algorithm of the src rrt-algorithm package is unreliable sinc eit depended on the discretization resolution of each edge. The algorithm checked if the points on the line at given interval do not lie in any obstacles. This was very unreliable as the ine segment wasn't checked as a whole entity. Hence the path returned by this package was checked again by Liang-Barsky algorithm (click to see the link).

## IV. Results

### A. Quality of computed paths

The paths evaluated by weighted A* algorithm were not optimal as the optimality depends on the parameter $\epsilon$. For majority of the scenarios the value of $\epsilon$ was chosen to be 3. Hence the algorithm gave 3-suboptimal path.
Also the paths computed were not post-processed and smoothened hence they have sharp turns.

TABLE I
PATH LENGTH COMPUTED BY THE ALGORITHMS

| Maps | Path length | |
|---|---|---|
| | ***Weighted A**** | ***Sampling-based*** |
| cube | 7.84 | 8.41 |
| window | 26.96 | 26.67 |
| tower | 33.94 | 30.125 |
| monza | 76.27 | 73.66 |
| flappy bird | 28.13 | 28.52 |
| room | 11.82 | 16.54 |
| maze | 78.95 | 94.01 |

### B. Efficiency of the algorithms

TABLE II
COMPUTATION TIME

| Maps | Time taken (seconds) | | |
|---|---|---|---|
| | ***Weighted A**** | ***Bi-RRT*** | ***RRT**** |
| cube | 7.49 | 1.19 | 0.6 |
| window | 5.14 | 61.31 | 24.67 |
| tower | 50.26 | 446.16 | 290.73 |
| monza | 661.49 | 877.239 | - |
| flappy bird | 65.93 | 113.93 | 107.64 |
| room | 15.01 | 3.46 | 877.239 |
| maze | - | 2684.47 | - |

### C. Path plots

*1) Weighted A* Plots:*
*2) RRT* algorithm paths:*
*3) Bi-Directional RRT:*

TABLE III
NUMBER OF CONSIDERED NODES

| Maps | Nodes explored | | |
|---|---|---|---|
| | ***Weighted A**** | ***Bi-RRT*** | ***RRT**** |
| cube | 24 | 20 | 347 |
| window | 164 | 2081 | 2081 |
| tower | 2580 | 3904 | 11792 |
| monza | 36512 | 25037 | - |
| flappy bird | 6984 | 1288 | 4481 |
| room | 1114 | 95 | 1840 |
| maze | - | 60118 | - |

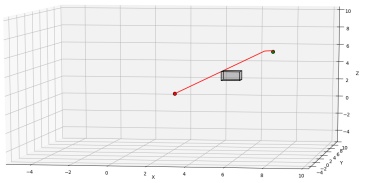

Fig. 1. Weighted A* Path in the single cube environment

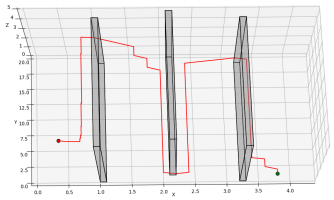

Fig. 2. Weighted A* Path in the monza environment

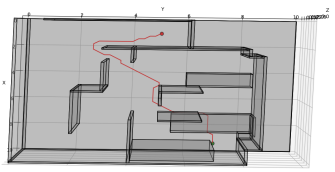

Fig. 3. Weighted A* Path in the room environment

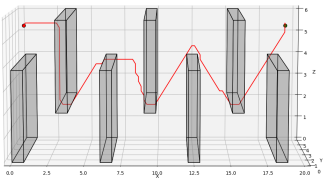

Fig. 4. Weighted A* Path in the flappy bird environment

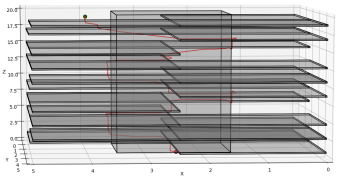

Fig. 5. Weighted A* Path in the tower environment

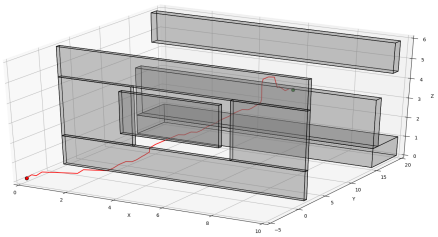

Fig. 6. Weighted A* Path in the window environment

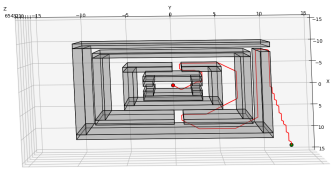

Fig. 7. Weighted A* Path in the maze environment

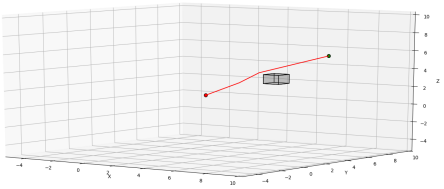

Fig. 8. RRT* Path in the single cube environment

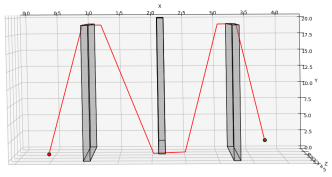

Fig. 9. RRT* Path in the monza environment

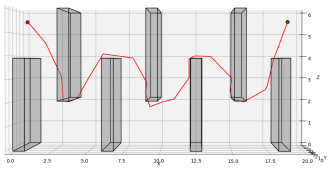

Fig. 10. RRT* Path in the flappy bird environment

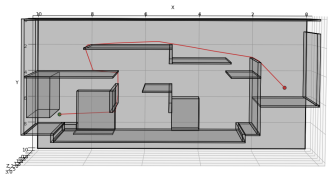

Fig. 11. RRT* Path in the room environment

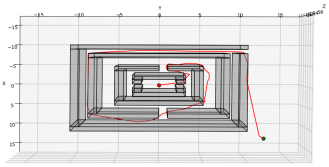

Fig. 12. RRT* Path in the maze environment

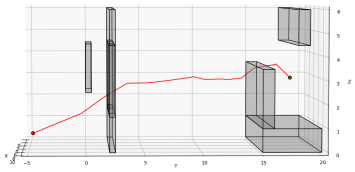

Fig. 13. RRT* Path in the window environment

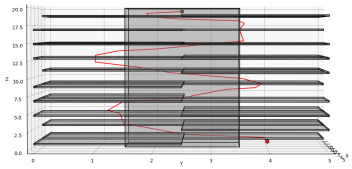

Fig. 14. RRT* Path in the tower environment

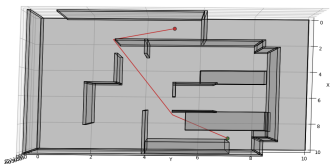

Fig. 15. Bi-Directional RRT* Path in the room environment

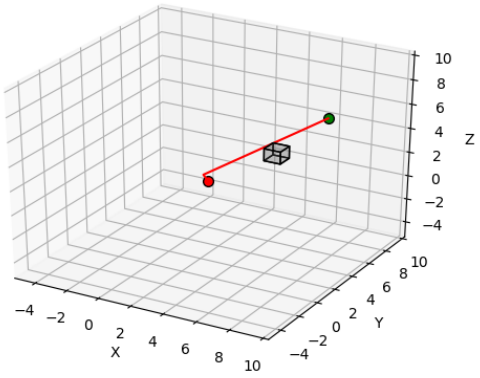


Fig. 16. Bi-Directional RRT* Path in the single cube environment

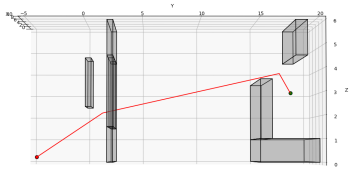

Fig. 17. Bi-Directional RRT* Path in the window environment

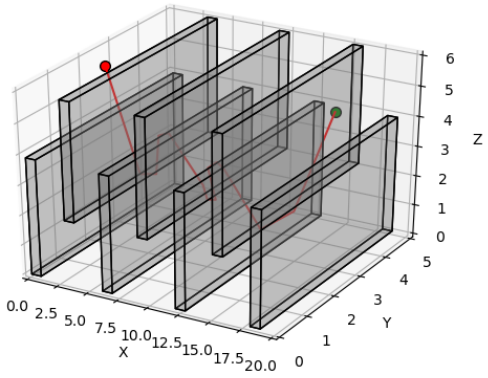


Fig. 18. Bi-Directional RRT* Path in the flappy bird environment

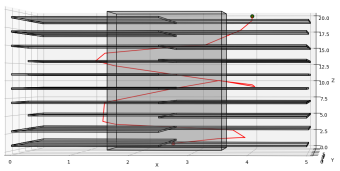

Fig. 19. Bi-Directional RRT* Path in the tower environment