Optimal Path Planning for an Omnidirectional Robot

Final Project Report

Mobina Amrollahi

Abstract

Rapidly-exploring Random Trees (RRT) have been widely used for over a decade and possess theoretical guarantees such as probabilistic completeness. However, under mild technical conditions, the cost of the solution returned by these algorithms converges almost surely to a non-an optimal value. Hence, the asymptotically optimal version of this algorithm, namely, RRT, is formulated and developed so that the cost of the returned solution converges almost surely to the optimum. Although the completeness guarantees are weaker, the efficiency and ease of im- implementation of these methods have bolstered interest in applying motion planning algorithms to various applications.*

1 Project Description

Consider a circular, omnidirectional mobile robot of radius $r = 10$ operating in a 2D world. The robot can move in the following way:

- **Translation** $T(d)$: The robot can move along the direction specified, where $d \in \mathbb{R}$ is the distance parameter. A positive distance $d > 0$ indicates forward movement, while a negative distance $d < 0$ indicates backward movement.

The workspace, where the robot operates, is a rectangular area defined by the set $[0, 300] \times [0, 300]$, with the bottom-left corner at $(0, 0)$ and the top-right corner at $(300, 300)$. Each location in the world is identified by a coordinate (x, y), where $x \in [0, 300]$ represents the horizontal position (x-axis) and $y \in [0, 300]$ represents the vertical position (y-axis).
The world contains a set $O = \{o_1, o_2, o_3\}$ of circular obstacles. Each circular obstacle is defined by a tuple $o = (x_o, y_o, r_o) \in \mathbb{R}^2 \times \mathbb{R}_{>0}$, where (x_o, y_o) is the position of the obstacle's center and r_o is its radius. In particular,

$$\begin{aligned} o_1 &= (200, 200, 30) \\ o_2 &= (75, 180, 50) \\ o_3 &= (180, 150, 20) \end{aligned}$$

In addition, the area outside the workspace boundaries must be considered as an obstacle. Assume, the robot's initial configuration is $q_I = (50, 50)$ and its goal configuration $q_G = (260, 260)$. The robot is considered to have reached the goal as soon as its center lies in any point of the circular goal region centered at $(260, 260)$, with a radius $r = 30$. The objective is to determine a sequence of the robot's center positions:

$$p = (x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n),$$

Where $(x_0, y_0) = (50, 50)$ and $(x_n, y_n) \in [230, 290] \times [230, 290]$, such that the path along the sequence of line segments $l_1, l_2, \ldots, l_n$, with each segment l_i connecting (x_{i-1}, y_{i-1}) and (x_i, y_i), is optimal. Additionally, the sequence of line segments corresponding to p must ensure that the robot remains entirely within the workspace and avoids collisions with any obstacles throughout its motion.

2 Problem Formulation

Configuration Space $\mathcal{C}$

Each configuration of the robot can be represented by (x, y), where $(x, y) \in \mathbb{R}^2$ specifies the position of the robot's center in the world frame. Thus, its configuration space is homeomorphic to $C = \mathbb{R}^2$. This group represents the set of all possible translations in a 2D plane.

Configuration Space Obstacle $\mathcal{C}_{\text{obs}}$

Consider the two following sets of obstacles:

1. $\mathcal{C}_{\text{obs,o}}$ corresponding to each circular obstacle $o = (x_0, y_0, r_0)$. The obstacle region $\mathcal{O}_o$ corresponding to a circular obstacle $o = (x_o, y_o, r_o)$ is given by

$$\mathcal{O}_o = \{(x', y') \in \mathbb{R}^2 \mid (x' - x_o)^2 + (y' - y_o)^2 \leq r_o^2\}.$$

Given a configuration $q = (x, y)$ of the robot, the region of the world occupied by the robot is given by

$$\mathcal{A}(q) = \{(x', y') \in \mathbb{R}^2 \mid (x' - x)^2 + (y' - y)^2 \leq r^2\}.$$

Hence, the configuration space obstacle corresponding to o is

$$\begin{aligned}\mathcal{C}_{\text{obs},o} &= \{(x, y) \in \mathbb{R}^2 \mid \mathcal{A}(x, y) \cap \mathcal{O}_o \neq \emptyset\},\\ &= \{(x, y) \in \mathbb{R}^2 \mid (x - x_o)^2 + (y - y_o)^2 \leq (r + r_o)^2\}.\end{aligned}$$

Now, by replacing the corresponding values of each obstacle, we have:

$$\begin{aligned}\mathcal{C}_{\text{obs},1} &= \{(x, y) \in \mathbb{R}^2 \mid \mathcal{A}(x, y) \cap \mathcal{O}_o \neq \emptyset\},\\ &= \{(x, y) \in \mathbb{R}^2 \mid (x - 200)^2 + (y - 200)^2 \leq (10 + 30)^2\}.\end{aligned}$$

$$\begin{aligned}\mathcal{C}_{\text{obs},2} &= \{(x, y) \in \mathbb{R}^2 \mid \mathcal{A}(x, y) \cap \mathcal{O}_o \neq \emptyset\},\\ &= \{(x, y) \in \mathbb{R}^2 \mid (x - 75)^2 + (y - 180)^2 \leq (10 + 50)^2\}.\end{aligned}$$

$$\begin{aligned}\mathcal{C}_{\text{obs},3} &= \{(x, y) \in \mathbb{R}^2 \mid \mathcal{A}(x, y) \cap \mathcal{O}_o \neq \emptyset\},\\ &= \{(x, y) \in \mathbb{R}^2 \mid (x - 180)^2 + (y - 150)^2 \leq (10 + 20)^2\}.\end{aligned}$$

2. $\mathcal{C}_{\text{obs},w}$ the configuration space obstacle corresponding to the area outside the workspace.

$$\mathcal{C}_{\text{obs},w} = \{(x, y) \in \mathbb{R}^2 \mid x < 10 \text{ or } x > 290 \text{ or } y < 10 \text{ or } y > 290\}.$$

The inequalities have to be strict because the workspace is closed.
Therefore, the complete configuration space obstacle region is therefore:

$$\mathcal{C}_{\text{obs}} = \mathcal{C}_{\text{obs},w} \cup \bigcup_{o=1}^{3} \mathcal{C}_{\text{obs},o}.$$

Free Configuration Space $\mathcal{C}_{\text{free}}$

$\mathcal{C} \setminus \mathcal{C}_{\text{obs}}$ is an open set. We will denote the obstacle-free space as $\mathcal{C}_{\text{free}} = \text{cl}(\mathcal{C} \setminus \mathcal{C}_{\text{obs}})$, where $\text{cl}(\cdot)$ denotes the closure of a set. This will result in $\mathcal{C}_{\text{free}}$ being a closed set, which is a crucial condition for RRT* to be asymptotically optimal. Therefore,

$$\begin{aligned}\mathcal{C}_{\text{free}} = \{(x, y) \in \mathbb{R}^2 \mid\, & x \in [10, 290]\\ \text{and} \quad & y \in [10, 290]\\ \text{and} \quad & (x - x_o)^2 + (y - y_o)^2 > (r + r_o)^2, \ \forall o \in O\}\end{aligned}$$

Set U of Actions

Since the robot has a radius of 10, it can move along each axis within the range $[0, 280]$, ensuring that the robot does not exceed the world boundaries, which are defined by a 300 by 300 grid. The robot must stay within these limits to avoid going outside the workspace. Hence, the set of actions is defined as:

$$U = \{(\delta x, \delta y) \in \mathbb{R}^2 \mid \delta x, \delta y \in \{-1, 0, 1\},\ 0 \leq \delta x \leq 280,\ 0 \leq \delta y \leq 280\}$$

Action Space $U(x)$

The action space for each configuration $(x, y) \in \mathcal{C}$ is defined as:

$$U(x) = \{u \in U \mid f((x, y), u) \in \mathcal{C}_{\text{free}}\}$$

State Transition Function f

$$f((x, y), u) = (x + u_1, y + u_2), \quad \forall (x, y) \in \mathcal{C}, \quad u = (u_1, u_2) \in U(x).$$

Time

Time is implicit, and it reflects the fact that the proper sequence of action must be followed [4].

Optimal Path Planning

Whether one is interested in minimizing total cost, time, or energy in a path planning problem, all these objectives concern the quality of the path returned by a motion planning algorithm, making optimal motion planning crucial [5]. However, until the discovery of RRT*, optimal sampling-based motion planning algorithms relied heavily on heuristics [3, 6, 1]. Karaman and Frazzoli have provided a systematic and thorough analysis, demonstrating how RRT* is probabilistically complete, asymptotically optimal, and computationally efficient compared to its standard counterpart RRT [2]. Therefore, developing such algorithms for motion planning problems is essential for achieving high-quality, optimal solutions.

Cost Function

In this report, the cost function is the Euclidean path length, which measures the straight-line distance traveled by the robot from one point to another and is used to evaluate the optimality of solution paths. Hence, $\texttt{Cost} : V \to \mathbb{R}_{\geq 0}$ maps a vertex $v \in V$ to the cost of the unique path from the root of the tree to v. Hence, the additive cost function is:

$$\texttt{Cost}(v) = \texttt{Cost}(\texttt{Parent}(v)) + c(\texttt{Line}(\texttt{Parent}(v), v))$$

For convinience, if $v_0 \in V$ is the root vertex of G, then $\texttt{Cost}(v_0) = 0$.

Assumptions

In this project, no differential constraints are considered. The dynamics are neglected and we're treating the system as if it has no inertia or acceleration. Rather, we're only considering the geometric path as if the robot is moving instantaneously between points without any physical limitations based on its own dynamics. Additionally, for simplicity, the samples are assumed to be drawn from a uniform distribution and are independent and identically distributed (i.i.d.). The distance function used to calculate the nearest neighbors is Euclidean distance. Finally, the obstacle locations, the initial configuration, and the goal region are known.

3 Approaches

In this part, the RRT* algorithm is outlined. This algorithm takes $(\mathcal{C}, q_I, q_G, \text{goal region radius}, \mathcal{O}, n, \eta)$, where $\mathcal{C}$ is the configuration space, q_I and q_G are the initial and goal configurations, goal region radius is the radius centered around q_G which is 30, $\mathcal{O}$ is the set of obstacles, $n \in \mathcal{N}$ is the number of iterations, and finally a prespecified value $\eta > 0$ which will be used in the steering function and determining the radius of a ball used to pick which vertices should be considered for connection. This algorithm will return a graph $G = (V, E)$, where $V \subset \mathcal{C}_{\text{free}}$, $\text{card}(V) \leq N + 1$, and $E \in V \times V$ as well as a path, which is the shortest path to the goal region starting from q_I.

3.1 Algorithms

We will be implementing the RRT* algorithm proposed by Karaman, and Emilio Frazzoli [2], shown in Algorithm 1. This algorithm adds points to the vertex set V considering connections from the new vertex x_{new} to vertices in X_{near}, i.e., other vertices that are within distance r where:

$$r(\text{card}(V)) = \min\{\gamma_{\text{RRT*}}(\log(\text{card}(V))/\text{card}(V))^{1/d}, \eta\}.$$

Where $\gamma_{\text{RRT}^*} > \left(2\left(1+\frac{1}{d}\right)\right)^{1/d}\left(\frac{\mu(\mathcal{C}_{\text{free}})}{\zeta_d}\right)^{1/d}$. d is the dimension of the space world, $\mu(\mathcal{C}_{\text{free}})$ denotes the Lebesgue measure (i.e., volume) of the free configuration space, and ζ_d is the volume of the unit ball in the d-dimensional Euclidean space. Since $\mu(\mathcal{C}) \geq \mu(\mathcal{C}_{\text{free}})$, to satisfy the inequality, we will be considering the following for γ_{RRT^*}:

$$\gamma_{\text{RRT}^*} = \left(2\left(1+\frac{1}{d}\right)\right)^{1/d}\left(\frac{\mu(\mathcal{C})}{\zeta_d}\right)^{1/d}$$

Hence, by plugging these values with respect to the problem statement, we have: γ_{RRT^*}:

$$\begin{aligned}
\gamma_{\text{RRT}^*} &= \left(2\left(1+\frac{1}{d}\right)\right)^{1/d}\left(\frac{\mu(\mathcal{C})}{\zeta_d}\right)^{1/d} \\
&= \left(2\left(1+\frac{1}{2}\right)\right)^{1/2}\left(\frac{300 \times 300}{\pi}\right)^{1/2} \\
&= \sqrt{3}\left(\frac{300}{\sqrt{\pi}}\right) \\
&\approx 293.16
\end{aligned}$$

,

4 Results

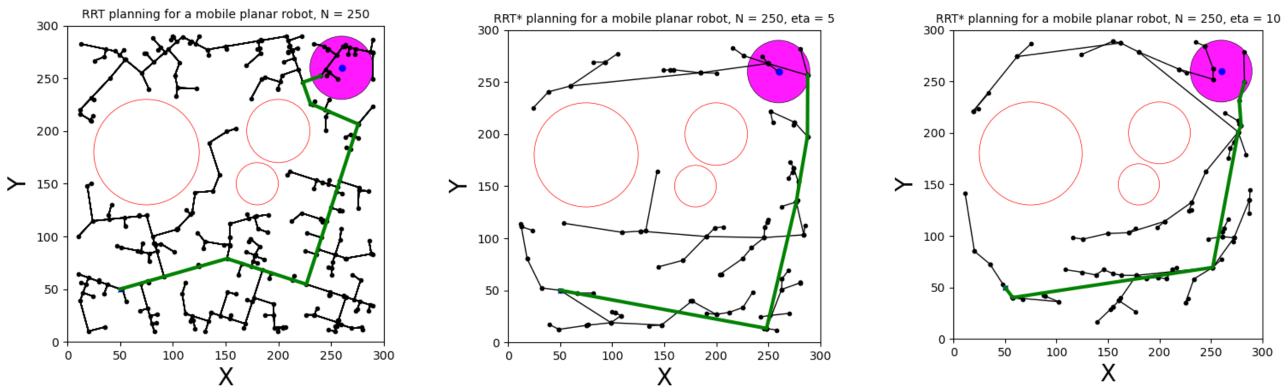
We ran RRT and RRT* for four different values of $N = \{250, 500, 1000, 1000\}$. Additionally, for RRT*, we overspecified two different values for $\eta = \{5, 10\}$ and ran RRT* with both of them. The results are shown in 1, 2, 3 and 4.

5 Discussions

As shown in images, RRT quickly finds feasible paths, but these are not guaranteed to be optimal. As the number of iterations increases, the path may change, illustrating that RRT is not asymptotically optimal.

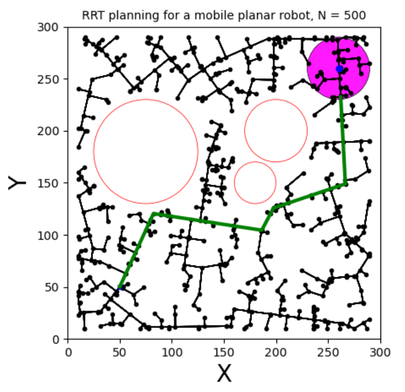
Algorithm 1: RRT*

```
Data: Sample $n$ points from $\mathcal{X}_{\text{free}}$, set the radius $r = \gamma_{\text{RRT*}}(\log(\text{card}(V))/\text{card}(V))^{1/d}$
Result: Graph $G = (V, E)$ with vertices and edges
$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ do
    $x_{\text{rand}} \leftarrow \text{SampleFree}_i$;
    $x_{\text{nearest}} \leftarrow \text{Nearest}(G = (V, E), x_{\text{rand}})$;
    $x_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
    if $\textit{ObstacleFree}(x_{nearest}, x_{new})$ then
        $X_{\text{near}} \leftarrow \text{Near}(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRT*}}(\log(\text{card}(V))/\text{card}(V))^{1/d}, \eta\})$;
        $V \leftarrow V \cup \{x_{\text{new}}\}$;
        $x_{\text{min}} \leftarrow x_{\text{nearest}}$; $c_{\text{min}} \leftarrow \text{Cost}(x_{\text{nearest}}) + c(\text{Line}(x_{\text{nearest}}, x_{\text{new}}))$;
        for each $x_{near} \in X_{near}$ do
            if $\textit{CollisionFree}(x_{near}, x_{new}) \wedge \textit{Cost}(x_{near}) + c(\textit{Line}(x_{near}, x_{new})) < c_{min}$ then
                $x_{\text{min}} \leftarrow x_{\text{near}}$; $c_{\text{min}} \leftarrow \text{Cost}(x_{\text{near}}) + c(\text{Line}(x_{\text{near}}, x_{\text{new}}))$;
            end
        end
        $E \leftarrow E \cup \{(x_{\text{min}}, x_{\text{new}})\}$;
        for each $x_{near} \in X_{near}$ do
            if $\textit{CollisionFree}(x_{new}, x_{near}) \wedge \textit{Cost}(x_{new}) + c(\textit{Line}(x_{new}, x_{near})) < \textit{Cost}(x_{near})$ then
                $x_{\text{parent}} \leftarrow \text{Parent}(x_{\text{near}})$;
                $E \leftarrow (E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\}$;
            end
        end
    end
end
return $G = (V, E)$
```

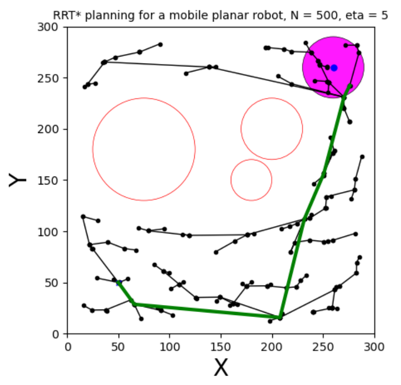


(a) RRT (b) RRT*, $\eta = 5$ (c) RRT*, $\eta = 10$

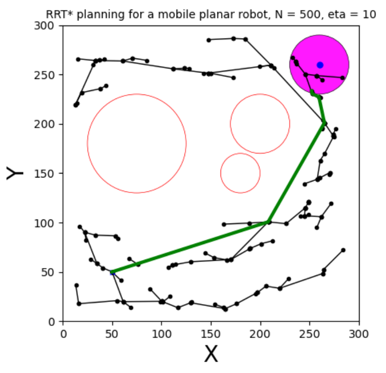
Figure 1: RRT and RRT* algorithms shown after 250 iterations.



(a) RRT

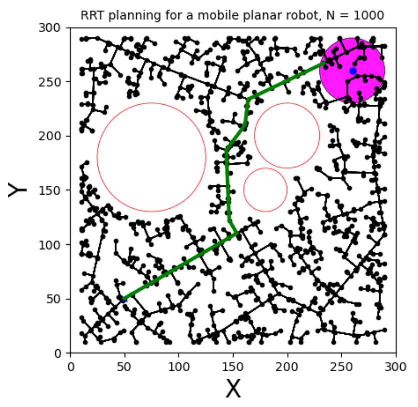


(b) RRT*, $\eta = 5$

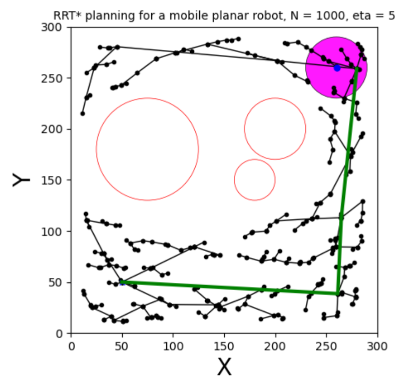


(c) RRT*, $\eta = 10$

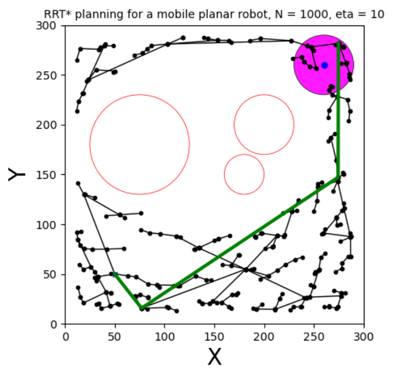
Figure 2: RRT and RRT* algorithms shown after 500 iterations.



(a) RRT

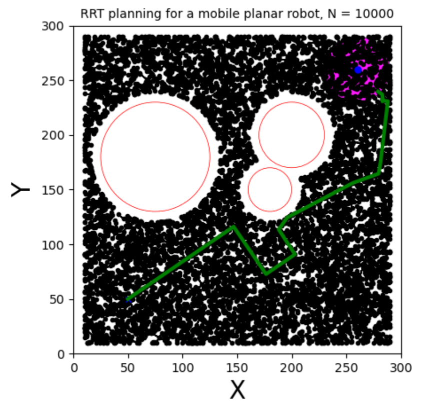


(b) RRT*, $\eta = 5$

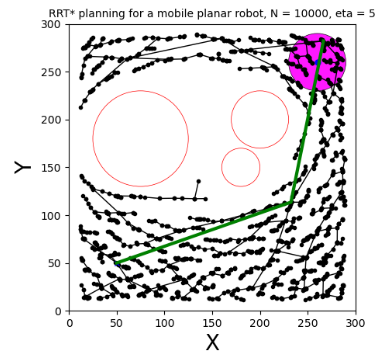


(c) RRT*, $\eta = 10$

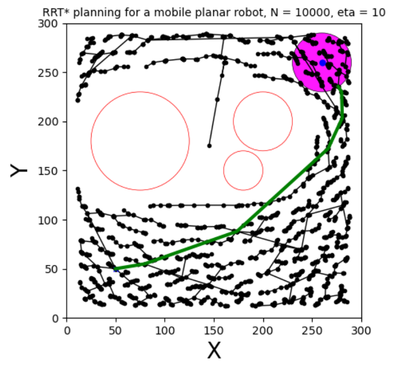
Figure 3: RRT and RRT* algorithms shown after 1000 iterations.



(a) RRT



(b) RRT*, $\eta = 5$



(c) RRT*, $\eta = 10$

Figure 4: RRT and RRT* algorithms shown after 10000 iterations.

For example, compare 500 and 1000 iterations in figure 2 and 3. In contrast, RRT* produces smoother and more optimal paths as the number of iterations and parameter tuning (such as η) increases. Unlike RRT, the paths found by RRT* become more consistent across iterations, with the solution approaching the optimal path as the number of samples grows, emphasizing its asymptotic optimality.
One key consideration when running RRT* is hyperparameter tuning. As seen in the images with different η values, the paths produced with a larger η are generally smoother and more optimal but may take longer to compute. Specifically, the radius of the ball around a sample point depends on the parameter η, which controls the step size when rewiring the tree. The radius function is given by:

$$r(\text{card}(V)) = \min\left\{\gamma_{\text{RRT*}}\left(\frac{\log(\text{card}(V))}{\text{card}(V)}\right)^{1/d}, \eta\right\}$$

Which can be rewritten as:

$$r = \frac{\min\left\{\gamma_{\text{RRT*}}\left(\frac{\log(\text{card}(V))}{\text{card}(V)}\right)^{1/d}, \eta\right\}}{\text{card}(V)}.$$

The term $\gamma_{\text{RRT*}}\left(\frac{\log(\text{card}(V))}{\text{card}(V)}\right)^{1/d}$ is a decreasing function. Since η is constant, the radius will follow a decreasing function at each iteration after the number of vertices exceeds a certain threshold. While the decreasing radius allows the RRT* algorithm to get closer to an optimal solution, it also makes the exploration process more computationally expensive. This suggests the importance of hyperparameter tunning of η.

6 Conclusion

In this work, we have explored the performance of RRT and RRT* algorithms for motion planning problems. While RRT efficiently finds feasible paths, it does not guarantee optimality, and the quality of the path can significantly improve with the number of iterations. In contrast, RRT* is designed to be asymptotically optimal, continuously refining the path with more iterations and samples.

References

[1] Robotics IEEE, Industrial Electronics and CORPORATE Automation Society Staff. *International Conference on Intelligent Robots and Systems*. IEEE Press, 1993.

[2] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[3] Yoshiaki Kuwata, Justin Teo, Gaston Fiore, Sertac Karaman, Emilio Frazzoli, and Jonathan P How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on control systems technology*, 17(5):1105–1118, 2009.

[4] SM Lavalle. Planning algorithms. *Cambridge University Press google schola*, 2:3671–3678, 2006.

[5] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

[6] Chris Urmson and Reid Simmons. Approaches for heuristically biasing rrt growth. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 2, pages 1178–1183. IEEE, 2003.