Implementation and Evaluation of the FF-RRT* Path-Planning Algorithm for Mobile Robotics

Areeb Imran 1007215960
John Lewczuk 1006273809
Longyu Li 1005844268
Mohammad Chaudhry 1006404810

Abstract: The paper explores and evaluates the FF-RRT* algorithm for mobile robotics path planning by seeking to replicate the results from the research paper [1] and comparing the performance of FF-RRT* against other Rapidly exploring Random Tree (RRT) algorithms. The FF-RRT* algorithm is a sampling-improved path planning method designed to handle complex environments, including concave cavity obstacles, and improve convergence rates compared to existing algorithms. FF-RRT* is a combination and expansion of the Fast-RTT* and F-RRT* algorithms. This paper replicates the results of the experiments performed in the FF-RRT* paper [1] and extends those results by demonstrating the performance of FF-RRT* on additional scenarios, including a floor plan and a map of the city of Mississauga.

Keywords: Mobile Robotics, Path-Planning, Rapidly Exploring Random Tree star (RRT*)

1 Introduction

Path planning is a crucial aspect of mobile robotics, especially in environments with obstacles of varying complexity. It involves the task of finding an optimal collision-free path from a starting point to a target destination within a predefined environment. Among the plethora of path planning algorithms used today, sampling-based approaches have gained significant traction, with Rapidly Exploring Random Trees (RRT) standing out in particular [2].

RRT benefits from excellent performance as it generates a feasible non-optimal path which does not involve complicated computation. RRT* extends the RRT algorithm, aiming to generate an optimal path [3]. The distinction lies in RRT*'s ability to systematically refine paths for improved optimality. Unfortunately, this optimality comes at the expense of slow convergence rates due to the increased computation complexity.

To address this issue, many RRT*-variants have been proposed in recent years to increase computation efficiency while maintaining path optimality. In this paper, we investigate three variants in particular: Informed RRT*, Fast-RRT*, and F-RRT*. Informed RRT* [4] works by restricting the sampling space once the first solution is found. The sampling space is restricted to an ellipsoidal subset of the state space such that all the nodes that can reduce the cost of the solution are within the ellipsoid. Fast-RRT* [5] adopts a hybrid sampling approach to narrow the sampling area and thus shorten the convergence rate. Nevertheless, this approach's applicability diminishes in complex environments with concave cavity obstacles.

On top of an improved sampling strategy, reducing the number of nodes is another effective way to further improve the optimization effect. A heuristic function is introduced to calculate the path cost [6] and to subsequently eliminate any point that fails to decrease the path cost. While this reduces the number of tree nodes, reducing the number of path nodes can also improve convergence rates. F-RRT* [7] leverages the *FindReachest* procedure with triangular inequality alongside the *CreatNode* procedure employing a dichotomy method to greatly reduce path costs. A balance is required for choosing the optimal value for the dichotomy parameter: reducing the value of $D_{dichotomy}$ increases

the quality of the path, while also increasing the computation time. Combining the sampling strategy algorithm with F-RRT* to improve the sampling quality can help compensate for this increasing computation time.

The FF-RRT* algorithm aims to address these challenges by combining the strengths and overcoming the weaknesses of Informed-RRT*, Fast-RRT* and F-RRT*. FF-RRT* introduces improvements in sampling strategies and path optimizations, with a focus on achieving shorter convergence rates without compromising path quality.

2 Related Work

The method proposed uses Informed RRT* as a baseline to compare the performance of the path planning algorithms derived from RRT* (Fast-RRT*, F-RRT*, and FF-RRT*). Also, the proposed method uses practical and unique simulation environments to evaluate the robustness and versatility of each algorithm. Most other method approaches compare the performance of the relevant path-planning algorithms relative to each other in basic simulation environments.

The Fast-RRT* and F-RRT* algorithms incorporate the guiding principles of RRT* and improve upon them to enhance performance. Subsequently, FF-RRT* embodies both Fast-RRT* and F-RRT*. Note that this section borrows from the related works of the paper [1].

2.1 Informed-RRT*

Informed RRT* is a simple modification to RRT* that improves the convergence rate and the final quality of the solution. Informed RRT* performs the same as RRT* until the first path to the goal is found. At this point, it no longer draws samples uniformly from the state space, but instead limits the sampling space to only the set of nodes that could increase the solution quality. The new sampling process samples a state from the unit ball centered on the midpoint between the goal and end states and then transforms this state using a rotation matrix, from the hyper-ellipsoid frame which has the start and end states as its focal point to the world frame. In effect, the sample space is now limited to this new ellipsoid which guarantees that for any state sample from the ellipsoid, the optimal path going through this state is less than the cost of the currently recorded optimal path. Figure 1 shows the pseudo-code of the main function, other details are outlined in [4].

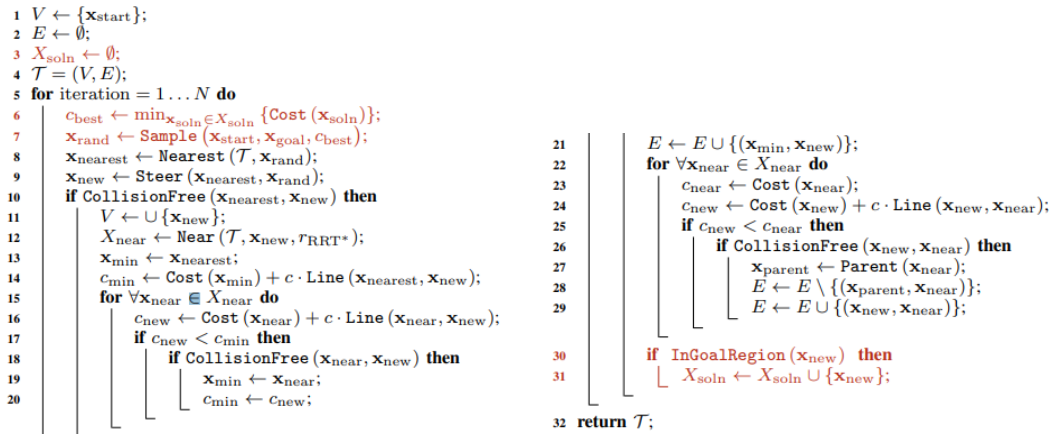


Figure 1: Informed-RRT*

2.2 Fast-RRT*

Fast-RRT* is one of FF-RRT*'s constituents and improves upon RRT* in two respects. First, it uses hybrid sampling to select the random node, in contrast to the random sampling method used by RRT. The sampling method is refined by applying the goal bias strategy and constrained sampling.

The sampling process uses the previously sampled node and a constant value λ, where $0 < \lambda < 1$. Second, it uses the *Backtracking* procedure to enhance the *ChooseParent* method. The ancestor nodes of the random node are determined by tracking from the temporary parent node to the starting node. The ancestor node with the lowest path cost then replaces the parent node of the random node. Figure 2 shows the pseudo-code of the main function, other details are outlined in [1].

Algorithm 1 Fast-RRT*

Input: x_s, x_g, X_{obs}, R_{near}, $n_{\max}$, λ

Output: $H=(Q,L)$

$Q \leftarrow \{x_s\}, L \leftarrow \varnothing$;
for $i=1$ to $n_{\max}$ do
  $x_{rand} \leftarrow Hybridsampling(x_g, x_{p\_rand}, \lambda)$;
  $X_{near} \leftarrow Near(V, x_{rand}, R_{near})$;
  $x_{parent} \leftarrow$ *improved ChooseParent*(x_{rand}, X_{near});
  $Q \leftarrow Q \cup \{x_{rand}\}$;
  $L \leftarrow L \cup \{(x_{rand}, x_{parent})\}$;
  if $CollisionFree(x_{rand}, x_g, X_{obs})$ then
    Return $H=(Q,L)$
  end if
end for
Return $H=(Q,L)$

Figure 2: Fast-RRT*

2.3 F-RRT*

F-RRT* is the other component of FF-RRT* and also carries two improvements over RRT. To begin with, the *FindReachest* procedure of F-RRT* builds upon the *ChooseParent* method the same way Fast-RRT* does with its *Backtracking* procedure. Therefore, the improved *ChooseParent* is effectively the same between F-RRT* and Fast-RRT*. Then, F-RRT* establishes the *CreatNode* procedure that creates a new parent node based on the dichotomy method to assist in finding an optimal path. Figure 3 shows the pseudo-code of the main function, other details are outlined in [1].

Algorithm 5 F-RRT*

Input: x_s, x_g, X_{obs}, R_{near}, $n_{\max}$, $D_{dichotomy}$

Output: $H=(Q,L)$

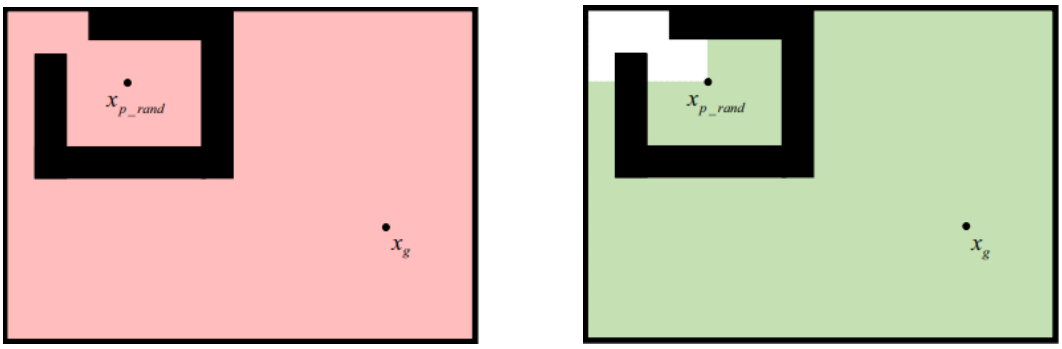
$Q \leftarrow \{x_s\}, L \leftarrow \varnothing$:
for $i=1$ to $n_{\max}$ do
  $x_{rand} \leftarrow SampleFree(i)$;
  $X_{near} \leftarrow Near(V, x_{rand}, R_{near})$;
  $x_{parent} \leftarrow$ *improved ChooseParent*(x_{rand}, X_{near});
  if $x_{parent} \neq x_s$ then
    $x_{anc} = Parent(x_{parent})$;
    $x_{parent} \leftarrow CreatNode(x_{rand}, x_{parent})$;
    $Q \leftarrow Q \cup \{x_{parent}, x_{rand}\}$;
    $L \leftarrow L \cup \{(x_{rand}, x_{parent})\} \cup \{(x_{parent}, x_{anc})\}$;
  else
    $Q \leftarrow Q \cup \{x_{rand}\}$;
    $L \leftarrow L \cup \{(x_{rand}, x_{parent})\}$;
  end if
  if $CollisionFree(x_{rand}, x_g, X_{obs})$ then
    Return $H=(Q,L)$;
  end if
  $H \leftarrow Rewire(H, x_{rand}, X_{near})$;
end for
Return $H=(Q,L)$

Figure 3: F-RRT*

FF-RRT* unites the advantages and conquers the flaws of F-RRT* and Fast-RRT*. Firstly, an improvement upon the hybrid sampling method is made by applying random sampling and constrained sampling. Secondly, the *ChooseParent* and *Rewire* procedures are adjusted to include the ancestor of the temporary parent node. These modifications emphasize sampling quality, fast convergence rates, and versatility to amplify the efficiency of the RRT* algorithm.

3 Methodology

FF-RRT* was the main algorithm that was replicated and evaluated in this study. It improves F-RRT* and Fast-RRT*, taking advantage of each algorithm's strengths to adopt a more efficient algorithm that creates more optimal paths and converges faster, particularly for concave environments. The distinct features of FF-RRT* come from its functions of *Improved HybridSampling*, *Improved ChooseParent*, *CreatNode*, and *Improved Rewire*.



(a) Random Sampling (b) Hybrid Sampling

Figure 4: Comparison of sampling area and hybrid sampling area taken from paper [1]. Illustrates the inherent weakness of hybrid sampling when applied to a concave cavity obstacle. (**a** illustrates the random sampling area in pink, **b** illustrates the hybrid sampling area in green)

To start, *Improved HybridSampling* improves on *HybridSampling* from Fast-RRT* by combining random sampling from *HybridSampling* with constrained sampling. Doing so allows *Improved HybridSampling* to better navigate concave obstacles as a chance of random sampling allows for FF-RRT* to escape such obstacles. In particular, we define a value λ, $0 < \lambda < 1$ which impacts the likelihood of random sampling compared to a constrained approach. Lower values result in more constrained sampling, which is considerably less efficient for concave obstacles while higher values provide a more random approach, allowing for more efficiency for concave obstacles. We set λ to 0.5, which allows FF-RRT* to handle concave obstacles quickly while maintaining efficiency during less blocking obstacles. Figure 5 shows the pseudocode for *Improved HybridSampling* from the original paper for FF-RRT* that was followed in our implementation [1].

Algorithm 9 Improved HybridSampling

Input: x_g, x_{p_rand}, λ
Output: x_{rand}
1 set $i=1$
2 $x_{rand} \leftarrow SampleFree(i)$;
3 $\lambda_r \leftarrow rand()$;
4 if $\lambda_r > \lambda$ then
5 while $abs\,|x_{rand} - x_g| > abs\,|x_{p\_rand} - x_g|$ do
6 $i = i+1$;
7 $x_{rand} \leftarrow SampleFree(i)$;
8 end while
9 Return x_{rand};
10 end if
11 Return x_{rand};

Figure 5: Improved HybridSampling

FF-RRT* uses the exact implementations of *Improved ChooseParent* and *CreatNode* from Fast-RRT* and F-RRT* respectively. *Improved Rewire* modifies the *Rewire* function from RRT* so that x_{near} is rewired to the parent of x_{rand} instead of x_{rand} as long as no obstacles are blocking x_{near}. This optimizes the path created by FF-RRT* by removing and not using sub-optimal nodes to allow for a shorter path that converges quicker than Fast-RRT* and F-RRT*. Figure 6 shows the pseudocode for *Improved Rewire* from the original paper for FF-RRT* that was followed in our implementation [1].

Algorithm 10 Improved Rewire

Input: H, x_{rand}, X_{near}

Output: H

1 for *each* $x_{near} \in X_{near}$ do

2 if $CollisionFree(Parent(x_{rand}), x_{near}, X_{obs})$ then

3 if $Cost(x_{near}) > Cost(Parent(x_{rand})) + Distance(Parent(x_{rand}), x_{near})$ then

4 $L \leftarrow (L \setminus \{(Parent(x_{near}), x_{near})\}) \cup \{(Parent(x_{rand}), x_{near})\}$;

5 end if

6 end if

7 end for

8 Return H

Figure 6: Improved Rewire

Algorithm 8 FF-RRT*

Input: x_s, x_g, X_{obs}, R_{near}, $n_{\max}$, $D_{dichotomy}$

Output: $H = (Q, L)$

1 $V \leftarrow \{x_s\}, E \leftarrow \varnothing$;

2 for $i = 1$ to $n_{\max}$ do

3 $x_{rand} \leftarrow improved\ HybridSampling(x_g, x_{p\_rand}, \lambda)$;

4 $X_{near} \leftarrow Near(V, x_{rand}, R_{near})$;

5 $x_{parent} \leftarrow improved\ ChooseParent(x_{rand}, X_{near})$;

6 if $x_{parent} \neq x_s$ then

7 $x_{anc} = Parent(x_{parent})$;

8 $x_{parent} \leftarrow CreatNode(x_{rand}, x_{parent})$;

9 $Q \leftarrow Q \cup \{x_{parent}, x_{rand}\}$;

10 $L \leftarrow L \cup \{(x_{rand}, x_{parent})\} \cup \{(x_{parent}, x_{anc})\}$;

11 else

12 $Q \leftarrow Q \cup \{x_{rand}\}$;

13 $L \leftarrow L \cup \{(x_{rand}, x_{parent})\}$;

14 end if

15 if $CollisionFree(x_{rand}, x_g, X_{obs})$ then

16 Return $H = (Q, L)$;

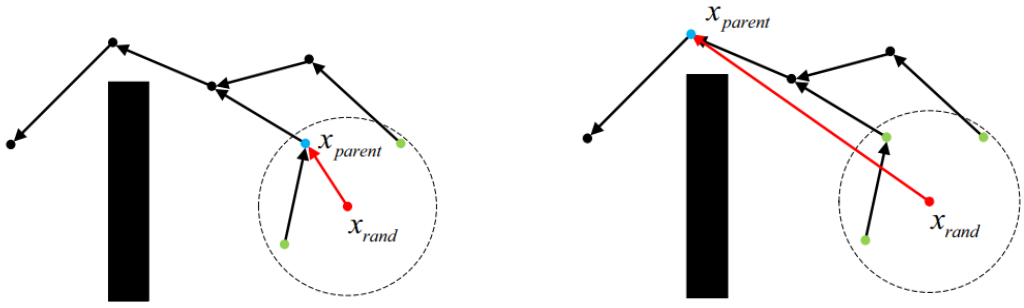
17 end if

18 $H \leftarrow improved\ Rewire(H, x_{rand}, X_{near})$;

19 end for

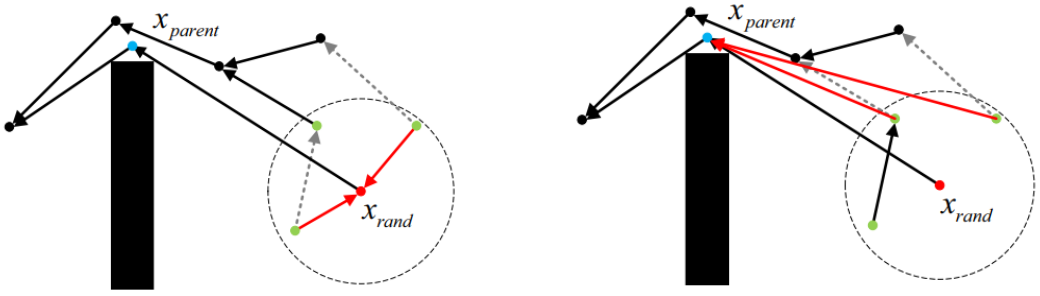
20 Return $H = (Q, L)$

Figure 7: FF-RRT* pseudocode



(a) RRT* (b) FF-RRT*

Figure 8: Comparison of *ChooseParent* and *improved ChooseParent* procedures taken from paper [1]. Illustrates the efficiency of the *improved ChooseParent* over the original *ChooseParent*. (**a** shows original *ChooseParent* of RRT*, **b** shows *improved ChooseParent* of FF-RRT*)



(a) RRT* (b) FF-RRT*

Figure 9: Comparison of *Rewire* and *improved Rewire* procedures taken from paper [1]. (**a** shows original *Rewire* of F-RRT*, **b** shows *improved ChooseParent* of FF-RRT*)

Lastly, the main implementation of FF-RRT* utilizes these improvements while building on F-RRT* main's function. Figure 7 shows the pseudocode of the main function from the original paper for FF-RRT* that was followed in our implementation [1].

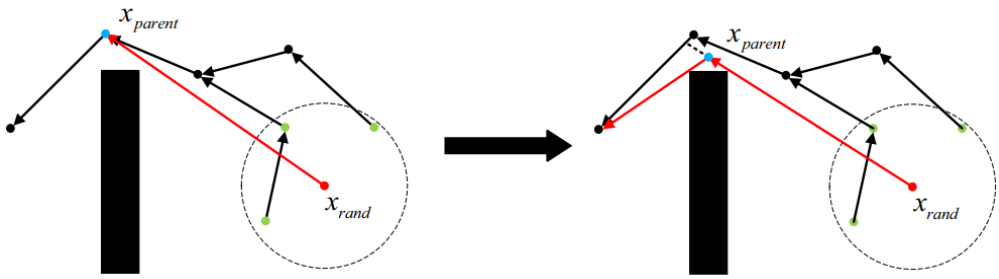


Figure 10: Taken from paper [1]. Illustrates the *CreatNode* procedure which remains unchanged from F-RRT*

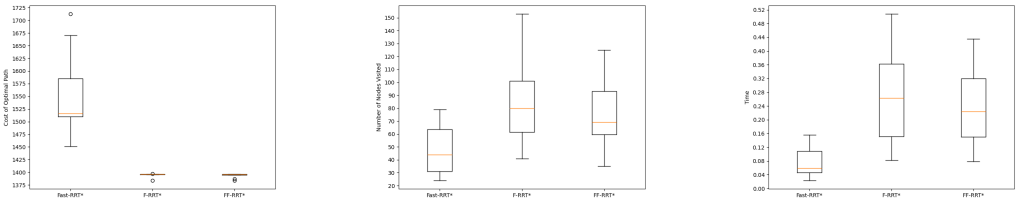
4 Evaluation

FF-RRT* was evaluated across a variety of 2-dimensional maps featuring different types of obstacles, including Simple Maze, Complex Maze, Complex Maze with Concave Cavity Obstacle, Cluttered, and additional custom maps with varying complexities. These included maps with regular shaped obstacles, irregular shaped obstacles, a narrow maze, a floor plan of a simple apartment, and a simplified map of Mississauga. The maps can be found as images in the worlds folder of the repository.

The performance comparison involved Fast-RRT*, F-RRT*, and FF-RRT* algorithms, each repeated 15 times on most maps and 10 times on the Mississauga map due to its size and complexity. In addition, the algorithms were compared with Informed-RRT* to only assess path optimality, as Informed-RRT* was significantly less efficient.

During the simulations, the following parameters were utilized:

- Maximum iterations for Fast-RRT*, F-RRT*, FF-RRT*: 30,000
- Maximum iterations for Informed-RRT*: 10,000 (except 20,000 for the Mississauga map)
- Maximum steering distance and radius: 70 pixels (except 150 pixels for the Mississauga map)
- Maximum distance to destination: 50 pixels (except 150 pixels for the Mississauga map)



(a) Cost of Optimal Path (b) Number of Nodes Visited (c) Computation Time

Figure 11: Statistical Results of Fast-RRT*, F-RRT*, FF-RRT* in a Simple Maze Environment

- $D_{dichotomy}$: 2 (except 100 for the Mississauga map). The dichotomy parameter makes a trade-off between convergence rate and optimality. A greater dichotomy converges to optimal solutions quicker but risks a sub-optimal solution with a smaller search space. A smaller dichotomy reinforces exploration and leads to an expansive search space, however, it will be more computationally exhausting to converge to the optimal solution. The dichotomy is set to 2 to ensure a potentially optimal solution, with 150 for the Mississauga map.
- Hybrid Lambda (λ): 0.5, which was kept consistent with the value from the research paper [1] (except for the concave complex maze, where 0.2 was used for Fast RRT* due to convergence issues).

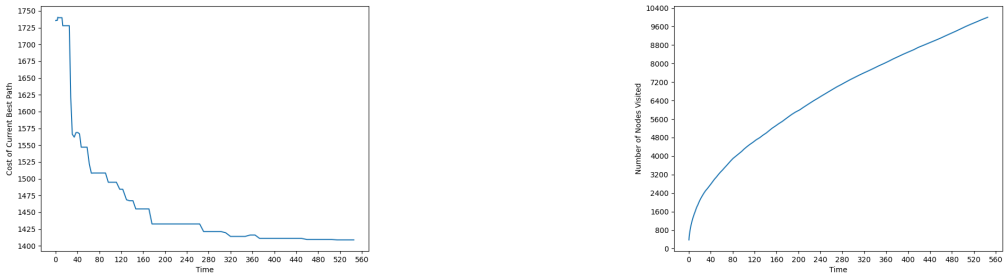
The simulation, conducted on an Intel Core i5-12600K with 16 GB of RAM, provided insights into the efficiency and optimality of path generation across different RRT* variants, which we will now discuss for each environment.

Note that we have not included the generated paths of each algorithm for each environment. However, those can be found in our code repository linked in our Appendix.

4.1 Simple Maze, Complex Maze, Complex Maze Concave, Cluttered

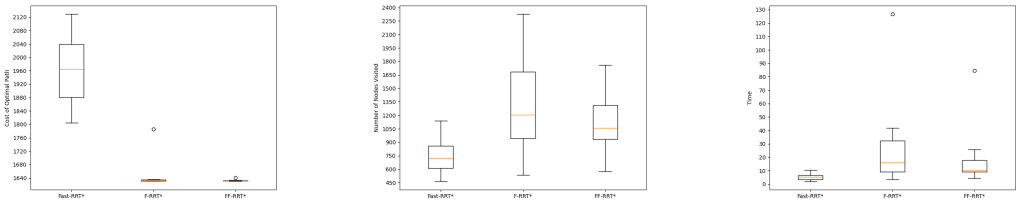
In the simple maze environment, the algorithms' performance and statistics in Figure 11 align closely with the findings of the FF-RRT* research paper [1]. Fast-RRT* exhibits significantly higher average path costs compared to both F-RRT* and FF-RRT*, indicating less optimal paths. This discrepancy showcases the efficacy of F-RRT* and FF-RRT* in generating more optimal paths, as evidenced by the path costs obtained by Informed-RRT* in Figure 12. Moreover, the lower average number of nodes in FF-RRT* compared to F-RRT* showcases the impact of the improved Hybrid Sampling procedure, enabling FF-RRT* to discover more optimal paths with a reduction of approximately 12% in the average number of nodes compared to F-RRT*, while also converging faster on average, by roughly 15%.

In the complex maze environment, Fast-RRT* continues to exhibit higher average path costs compared to F-RRT* and FF-RRT* as shown in Figure 13, highlighting the superior path optimality achieved by the latter two algorithms in navigating complex obstacles. This aligns with the observations from the FF-RRT* research paper [1]. Additionally, FF-RRT* demonstrates the capability to generate more optimal paths compared to Fast-RRT*, while also achieving faster convergence compared to F-RRT*. This is evidenced by a 12% reduction in the average number of nodes and a 15-20% reduction in the average computation time compared to F-RRT*. Also, the average computation time for FF-RRT* shows more stability than F-RRT*. Remarkably, both F-RRT* and FF-RRT* paths exhibit lower costs than those obtained by Informed-RRT* shown in Figure 14, despite significantly fewer nodes being visited (roughly 1000 nodes versus 10,000 nodes for Informed-RRT*), which shows the efficiency of FF-RRT* in finding optimal paths with reduced



(a) Cost of Optimal Path (b) Number of Nodes Visited

Figure 12: Statistical Results of Informed-RRT* in a Simple Maze Environment



(a) Cost of Optimal Path (b) Number of Nodes Visited (c) Computation Time

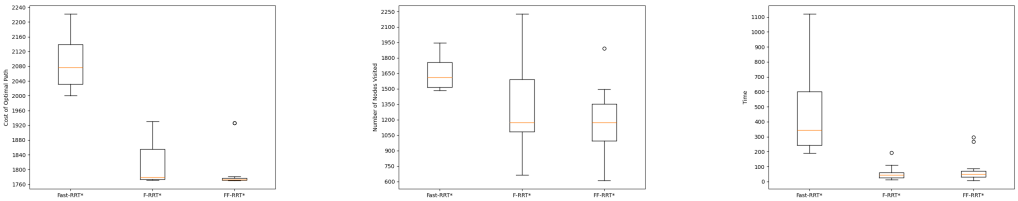
Figure 13: Statistical Results of Fast-RRT*, F-RRT*, FF-RRT* in a Complex Maze Environment

computational overhead for this environment.

In the concave complex maze environment, FF-RRT* demonstrates significantly more stable path costs compared to F-RRT*, despite their similar average cost values, shown in Figure 15. This stability extends to the number of nodes visited, reflecting FF-RRT*'s robust performance in navigating complex concave cavity obstacles. While both algorithms show comparable computation times, the consistent performance of FF-RRT* highlights its stability in challenging environments. Similar to the regular complex maze environment, both F-RRT* and FF-RRT* paths generate lower costs than those generated by Informed-RRT* shown in Figure 16, despite visiting significantly fewer nodes (around 1,200 nodes versus 9,200 nodes for Informed-RRT*). It should also be noted that the Fast-RRT* algorithm only converged 4 out of the 15 simulation results, indicating its limitations in handling concave cavity obstacles effectively.

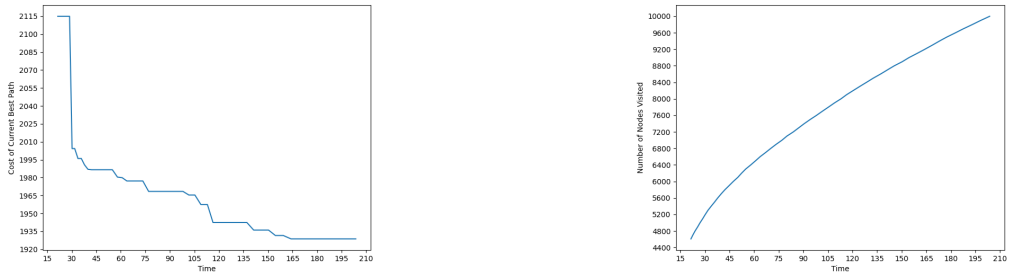
(a) Cost of Optimal Path (b) Number of Nodes Visited

Figure 14: Statistical Results of Informed-RRT* in a Complex Maze Environment



(a) Cost of Optimal Path (b) Number of Nodes Visited (c) Computation Time

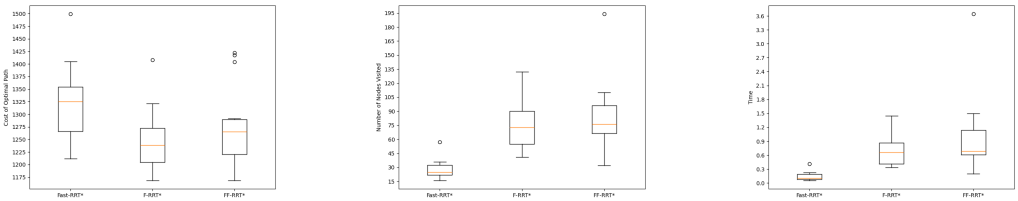
Figure 15: Statistical Results of Fast-RRT*, F-RRT*, FF-RRT* in a Complex Maze Environment with Concave Cavity Obstacle



(a) Cost of Optimal Path (b) Number of Nodes Visited

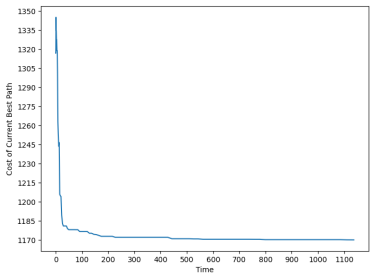
Figure 16: Statistical Results of Informed-RRT* in a Complex Maze Environment with Concave Cavity Obstacle

In the cluttered environment, our results deviated from those reported in the FF-RRT* research paper [1], as seen in Figure 17. F-RRT* exhibited a slightly lower average cost path compared to FF-RRT*, coupled with a more stable computation time. Although F-RRT* seemed to perform marginally better than the other algorithms in this context—achieving an average cost reduction of approximately 3-5% compared to FF-RRT*—the differences were minimal. Both algorithms showed similar average numbers of nodes visited and computation times. The limited simulation runs may have impacted result stability. Despite this, neither F-RRT* nor FF-RRT* produced optimal paths, as indicated by Informed-RRT*'s approximately 6% more optimal path, shown in Figure 18. However, the efficiency gains of F-RRT* and FF-RRT*, in terms of computation time and node visits, outweighed the marginal increase in path optimality observed with Informed-RRT*.

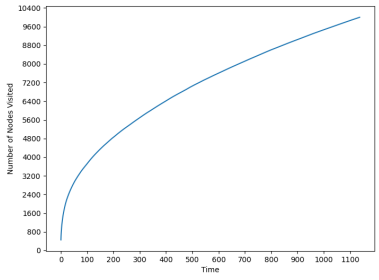


(a) Cost of Optimal Path (b) Number of Nodes Visited (c) Computation Time

Figure 17: Statistical Results of Fast-RRT*, F-RRT*, FF-RRT* in a Cluttered Environment

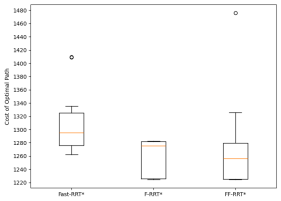


(a) Cost of Optimal Path

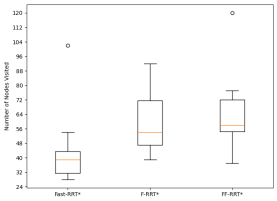


(b) Number of Nodes Visited

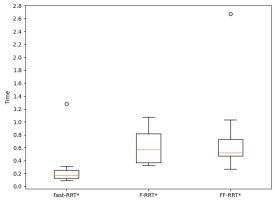
Figure 18: Statistical Results of Informed-RRT* in a Cluttered Environment



(a) Cost of Optimal Path



(b) Number of Nodes Visited

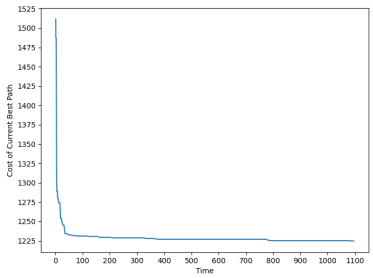


(c) Computation Time

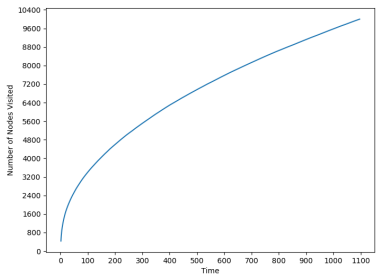
Figure 19: Statistical Results of Fast-RRT*, F-RRT*, FF-RRT* in an Environment with Regular Shaped Obstacles

4.2 Regular, Irregular, Narrow

Moving onto our custom environments, starting off with an environment containing regular shaped obstacles. The average path costs across the three algorithms were comparable, as seen in Figure 19. F-RRT* demonstrated a slight improvement of about 2-3% in average path cost over Fast-RRT*, with FF-RRT* showing a similar gain compared to F-RRT*. However, Fast-RRT* clearly stood out in terms of efficiency, showcasing 30% fewer average node visits and 66% less average computation time compared to F-RRT* and FF-RRT*. Despite Fast-RRT*'s efficiency, it still achieved nearly optimal path costs, as indicated by the path cost obtained by Informed-RRT* in Figure 20, which was approximately only 6% lower than Fast-RRT*'s average path cost. This highlights Fast-RRT*'s effectiveness in navigating environments with simple obstacles over FF-RRT*.

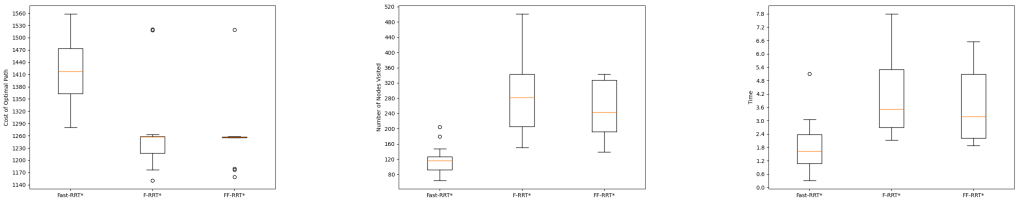


(a) Cost of Optimal Path



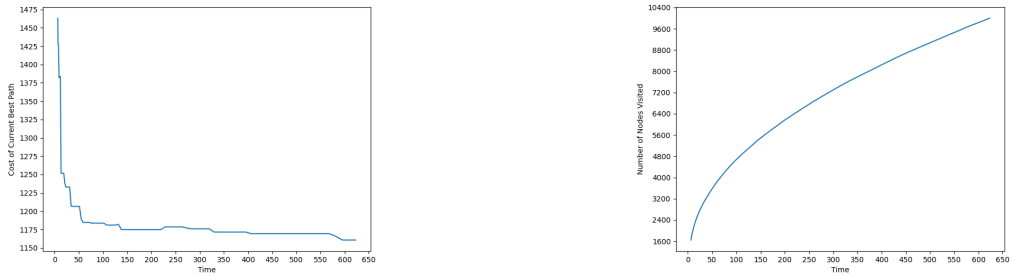
(b) Number of Nodes Visited

Figure 20: Statistical Results of Informed-RRT* in an Environment with Regular Shaped Obstacles



(a) Cost of Optimal Path (b) Number of Nodes Visited (c) Computation Time

Figure 21: Statistical Results of Fast-RRT*, F-RRT*, FF-RRT* in an Environment with Irregular Shaped Obstacles



(a) Cost of Optimal Path (b) Number of Nodes Visited

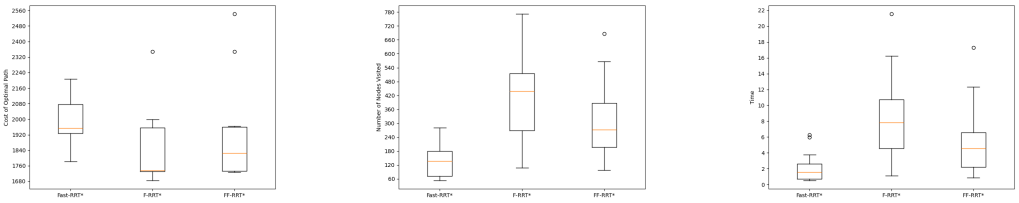
Figure 22: Statistical Results of Informed-RRT* in an Environment with Irregular Shaped Obstacles

These results do not directly translate to an environment with irregular shaped obstacles. F-RRT* and FF-RRT* showcase superior path cost optimization compared to Fast-RRT*, where Fast-RRT*'s average path cost surpasses the former two algorithms by 12%, shown in Figure 21. Despite this, Fast-RRT* demonstrates remarkable efficiency by visiting roughly half the nodes on average compared to FF-RRT*, coupled with approximately half the average computation time as well. This efficiency, however, comes at a cost of reduced path optimality, as evidenced by Informed-RRT*'s 17% more optimal path cost compared to Fast-RRT*, and about 7% more optimal path cost compared to F-RRT* and FF-RRT*, shown in Figure 22. While FF-RRT* maintains stable path costs overall, the efficiency advantage of Fast-RRT* remains notable, highlighting the trade-offs between path optimality and computational efficiency across different obstacle configurations.

In the narrow maze environment, F-RRT* achieves a more optimal average path cost compared to Fast-RRT*, although it is the least efficient among the three algorithms, shown in Figure 23. Despite F-RRT*'s average path cost being approximately 10% lower than Fast-RRT*'s, with Informed-RRT*'s path cost aligning closely with F-RRT* (Figure 24), Fast-RRT*'s efficiency shines through once again. It visits almost half the nodes that FF-RRT* does on average, and its computation time is also roughly half that of FF-RRT*. This emphasizes Fast-RRT*'s noteworthy efficiency even in challenging environments like narrow mazes.

4.3 Floor Plan

In the context of simulating a robot vacuum's path planning, we constructed a simple floor plan environment with two distinct target states. Across both scenarios (Figures 25 and 26, F-RRT* and FF-RRT* demonstrated very similar average path costs, albeit with F-RRT* exhibiting better stability in its results. Fast-RRT* showed roughly half the average number of nodes compared to F-RRT*, which in turn was about half of FF-RRT*'s node count. This pattern was consistent for average computation times as well. Both F-RRT* and FF-RRT* achieved nearly optimal



(a) Cost of Optimal Path (b) Number of Nodes Visited (c) Computation Time

Figure 23: Statistical Results of Fast-RRT*, F-RRT*, FF-RRT* in a Narrow Maze Environment

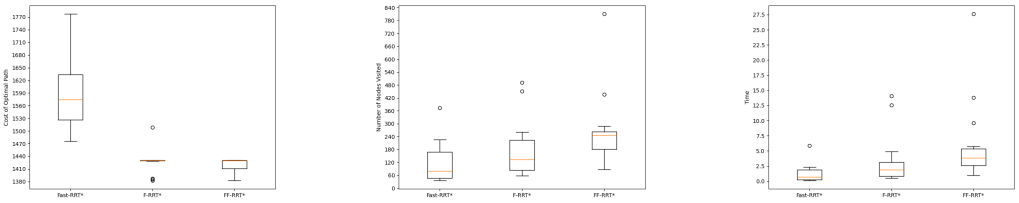
(a) Cost of Optimal Path (b) Number of Nodes Visited

Figure 24: Statistical Results of Informed-RRT* in a Narrow Maze Environment

path costs relative to Informed-RRT*, with Informed-RRT* (Figures 27 and 28) showcasing a path cost improvement of approximately 12% over Fast-RRT* across both scenarios. While F-RRT* outperformed FF-RRT* in this context, the efficiency of Fast-RRT* remains noteworthy, emphasizing the trade-offs between path optimality and computational efficiency across different planning scenarios.

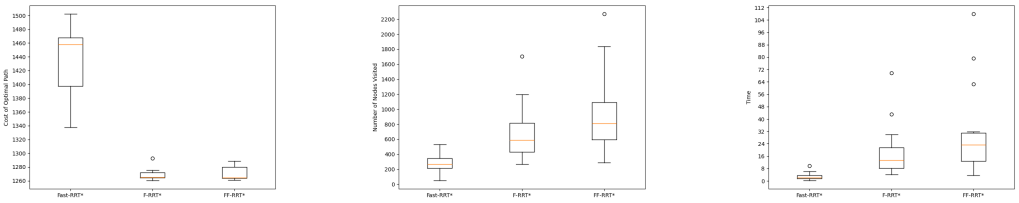
4.4 Mississauga

In our simulation on a larger scale using a simplified map of the city of Mississauga, Fast-RRT* and F-RRT* showcased similar average path costs, with FF-RRT* achieving a modest 4% improvement over both, seen in Figure 29. Despite this marginal gain, F-RRT* exhibited a notable advantage by visiting 25-30% fewer nodes on average compared to FF-RRT*, while Fast-RRT* showed a 40% reduction in node visits compared to F-RRT*. Moreover, F-RRT* demonstrated about a 30% reduction in computation time compared to FF-RRT*, while Fast-RRT* boasted a nearly halved average computation time compared to F-RRT*. However, the path cost generated by



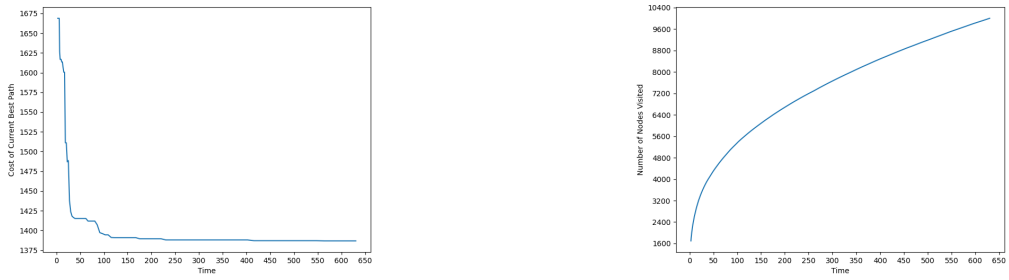
(a) Cost of Optimal Path (b) Number of Nodes Visited (c) Computation Time

Figure 25: Statistical Results of Fast-RRT*, F-RRT*, FF-RRT* in a Floor Plan Environment



(a) Cost of Optimal Path (b) Number of Nodes Visited (c) Computation Time

Figure 26: Statistical Results of Fast-RRT*, F-RRT*, FF-RRT* in a Floor Plan Environment



(a) Cost of Optimal Path (b) Number of Nodes Visited

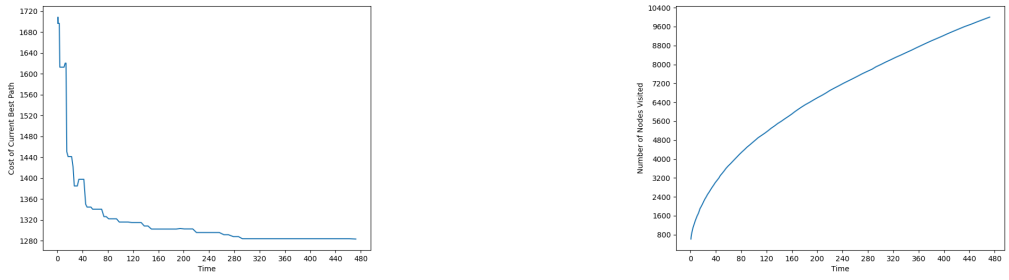
Figure 27: Statistical Results of Informed-RRT* in a Floor Plan Environment

Informed-RRT* was roughly 5% less than FF-RRT*, shown in Figure 30, a marginal improvement. While Fast-RRT* appears to be the more efficient choice for such a scaled environment, exploring different dichotomy values could help strike a balance between FF-RRT*'s path optimality and its computational efficiency.

5 Limitations

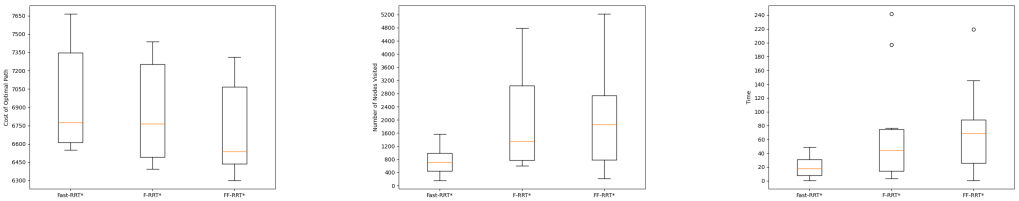
While FF-RRT* presents significant advancements in path planning efficiency and convergence rates, certain limitations and areas for future improvement are evident:

1. **Computation Time**: The increased computation due to the application of dichotomy remains a challenge, indicating the need for further optimization to reduce computational overhead.



(a) Cost of Optimal Path (b) Number of Nodes Visited

Figure 28: Statistical Results of Informed-RRT* in a Floor Plan Environment



(a) Cost of Optimal Path (b) Number of Nodes Visited (c) Computation Time

Figure 29: Statistical Results of Fast-RRT*, F-RRT*, FF-RRT* in the Mississauga City Environment

(a) Cost of Optimal Path (b) Number of Nodes Visited

Figure 30: Statistical Results of Informed-RRT* in the Mississauga City Environment

2. **Three-Dimensional Environments**: The applicability of FF-RRT* to three-dimensional environments is a valuable research direction, requiring adaptations to maintain efficiency and path continuity.
3. **Continuity in Path Generation**: For robots like drones, ensuring the continuity of generated paths is crucial, warranting additional research efforts in path smoothness and trajectory planning.

6 Conclusion

In conclusion, the FF-RRT* algorithm presents a novel approach to path planning that combines and improves upon the strengths of F-RRT* and Fast-RRT*. Its major contributions include:

- The introduction of the improved HybridSampling procedure, enhancing sampling quality and reducing convergence time.
- Overcoming the limitations of the original HybridSampling in Fast-RRT*, particularly in environments with concave cavity obstacles.
- Demonstrating superior performance in various environments, showcasing significant reductions in convergence time compared to F-RRT* and Fast-RRT*.

However, it is noteworthy that Fast-RRT* and F-RRT* outperformed FF-RRT* in a number of different environments, highlighting the trade-offs between path optimality and computational efficiency. Future research directions should focus on addressing the computational challenges, expanding applicability to three-dimensional spaces, and ensuring path continuity for practical robotic applications.

A Contributions

Contributions of each group member:

- **Areeb Imran**: I worked on implementing the F-RRT* algorithm and created a skeleton test script that was extended and refined to produce the appropriate plots for the results & evaluation section of the report. For the report, I worked on the related works section of the report (excluding the Informed RRT* section).
- **John Lewczuk**: I worked on implementing the Informed RRT* algorithm, the abstract, introduction, and Informed RRT* sections of the report.
- **Longyu Li**: I worked on implementing FF-RRT* and also took a road map that includes most of Mississauga from Google Maps API and processed it to work as a world for our algorithms. I also worked on the methodology part of the report.
- **Mohammad Chaudhry**: I worked on implementing Fast-RRT*, generated the regular, irregular, narrow and floor plan environments, and worked on the evaluations section of the report.

B Code Repository

The source code, as well as the paths and plots for all the algorithms in each environment can be found in our code repository.

References

[1] J. Cong, J. Hu, Y. Wang, Z. He, L. Han, and M. Su. FF-RRT*: a sampling-improved path planning algorithm for mobile robots against concave cavity obstacle. *Complex & Intelligent Systems*, 9(6):7249–7267, 2023. ISSN 2198-6053. doi:10.1007/s40747-023-01111-6. URL https://doi.org/10.1007/s40747-023-01111-6.

[2] S. M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998. URL https://api.semanticscholar.org/CorpusID:14744621.

[3] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011. doi:10.1177/0278364911406761. URL https://doi.org/10.1177/0278364911406761.

[4] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. Sept. 2014. doi:10.1109/iros.2014.6942976. URL http://dx.doi.org/10.1109/IROS.2014.6942976.

[5] Q. Li, J. Wang, H. Li, B. Wang, and C. Feng. Fast-RRT*: An improved motion planner for mobile robot in two-dimensional space. *IEEJ Transactions on Electrical and Electronic Engineering*, 17(2):200–208, 2022.

[6] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the RRT*. pages 1478–1483, 2011. doi:10.1109/ICRA.2011.5980479.

[7] B. Liao, F. Wan, Y. Hua, R. Ma, S. Zhu, and X. Qing. F-RRT*: An improved path planning algorithm with improved initial solution and convergence rate. *Expert Systems with Applications*, 184:115457, 2021.