Leveraging Opportunism in Sample-Based Motion Planning

Michael W. Lanighan and Oscar Youngquist

***Abstract*— Sample-based motion planning approaches, such as RRT*, have been widely adopted in robotics due to their support for high-dimensional state spaces and guarantees of completeness and optimality. This paper introduces an RRT* approach (ORRT*) that leverages opportunism to (1) find solutions quickly, (2) reduce wasted compute, and (3) improve data efficiency. The key insight of the approach is to make the most of compute when expanding the search tree by adding the last viable configurations found when connecting new nodes rather than rejecting the sampled nodes outright, allowing for more productive exploration of the space. We evaluate the proposed approach in a set of mobility and manipulator postural control domains, contrasting the performance of the opportunistic approach with state-of-the-art RRT* variants. Our analysis shows that such an approach has desirable characteristics and warrants further exploration.**

I. Introduction

Guarantees on probabilistic completeness and asymptotic optimality, along with support for high-dimensional state spaces have made RRT* approaches widespread in robotics. Although RRT* converges to the optimal solution in the limit, in practice only a finite number of additional samples can be made before accepting a solution. As such, work has focused on strategies to make sampling more efficient and propose techniques to minimize wasteful collision checks. This paper introduces an *opportunistic* RRT* variant (ORRT*) that leverages opportunism when building the tree to improve search behavior. Opportunistic strategies are proposed to (1) reduce wasted compute, (2) improve data efficiency, and (3) improve overall plan times and/or generated path lengths. The approach is evaluated in two domains: planar navigation and postural control for a 7 degree-of-freedom (DOF) manipulator.

II. Related Work

RRT approaches have been a mainstay of robotics planning for over a quarter of a century [1]. The effectiveness and simplicity of the approach led to widespread adoption in high degree-of-freedom robotics planning problems. RRT is not goal-directed—it is purely sample-based—so resulting plans can yield long, indirect paths. To address such optimality concerns, Karaman and Frazzoli introduced RRT* [2]. RRT* guides RRT progress by rewiring the search tree during planning in order to maintain shortest-paths when attaching new nodes. This effectively makes shortcuts between nodes in situ based on a distance heuristic. Algorithmically, RRT* is exactly the same as RRT except *Extend* is replaced with a new function *Extend**(shown in black in Algorithm 1), which rewires the tree if possible during search. The resulting paths are qualitatively better, with more goal-directed motion and are provably optimal at the limit [2].

Authors are with TRACLabs Inc. 1331 Gemini Street; Houston, TX 77058. Corresponding Author: mlanighan@tracblabs.com. This work was supported by NASA contract 80NSSC21C0622.

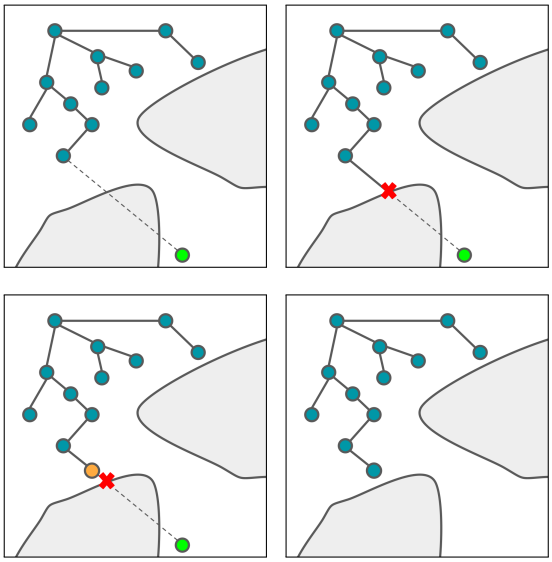


Fig. 1. Illustrating the opportunistic behavior of the planner. The tree attempts to connect to a sampled configuration (shown in green). When testing the path to the new node, a collision is detected midway (shown by the red X). In RRT* the work validating the motions up to the collision would be wasted. Instead, we opportunistically create a node β along the path from the detected collision (shown in orange). This new node is then added to the tree as if it were the original target.

In addition to rewiring, various attempts have been made in the field to improve the sampling behavior of RRT* [3], [4], [5]. One such promising solution, Informed-RRT* [6], constrains samples to the hyperellipsoid bounded by the best solution found so far. This bound limits samples to only those that may possibly improve the current solution. Such targeted sampling rapidly improves the quality of the solution compared to pure random sampling. In addition to modifying the sampling procedure, Informed-RRT* is achieved through modifying the RRT* *Extend** algorithm (black text) with the addition of the red lines shown in Algorithm 1. Hybrid RRT planners extend the application of Informed-RRT*, by initially searching as a bidirectional RRT to quickly find an initial path and then collapsing to a single tree to optimize the path found with informed sampling [7].

Despite the improvements conferred by RRT* and its variants, they still potentially waste compute cycles by performing wasteful collision checks when connecting new nodes to the tree. "Lazy" approaches have been proposed to overcome this by putting off expensive collision checks until an initial path is found and only then remove connections invalidated

by collisions [8]. However, as edges are removed, segments of the search tree may become disconnected, requiring either management of the resulting subtrees or implementation as a graph. Tahirovic and Ferizbegovic introduced Rapidly-exploring Random Vines (RRV) to address this wastefulness by projecting in-collision samples onto a hyper-plane created using statistics from an additional round of sampling concentrated around the point of collision [9]. Li and Chen introduce additional sampling heuristics (both for new nodes and for collision statistics) to improve the performance of RRV in highly cluttered environments, where the additional round of sampling can incur additional performance costs [10].

An additional well-known shortcoming of RRT-based planners is their dependence on the step-size parameter (η) used in the steering function. This parameter controls the trade-off between taking large steps in configuration space while building the tree—thereby reaching the goal faster—and taking smaller steps which encourages path optimality. The impact of this parameter is highly environmentally dependent. In open environments, a large η can rapidly cover the search space to reach the goal. In contrast, in cluttered and complex environments, the same η may abandon numerous samples due to environmental collisions, wasting compute and increasing search time. Consequently, selecting an appropriate η often involves time-consuming trial and error, adding to the difficulty of implementing RRT algorithms in practice.

To address this issue, methods have been proposed that enable dynamic η in RRT*. Wang and Heng eliminate η during planning by attempting to attach to the sampled node across its entire displacement from the nearest neighbor [11]. If a collision is found, instead of adding a previous configuration to the tree, the in-collision sample is added to a set of *CandidatePoints*, which they use to bias the sampling procedure away from obstacles. McCourt *et al.* increase η proportional to the number of nodes between the current node and the root of the tree [12]. An *et al.* replace the configuration space η with a maximum Cartesian displacement value set to the width of the smallest obstacle in the workspace. This value is then used to derive joint space displacement bounds based on the current configuration of the robot and the Cartesian displacement expected for a sampled joint displacement [13]. Approaches to dynamically update η based on the density of obstacles around the nearest-neighbor node have also been investigated [14], [15]. Al-Ansarry *et al.* also removes η by repeatedly adding a new node at the midpoint between the nearest-neighbor and the in-collision sample, until either the goal is reached or no more collision-free nodes can be added [16], similar to RRT-Connect [17], [18].

As a simpler alternative, we investigated an *opportunistic* strategy to improve planning performance. Traditionally, if a collision is detected during search in RRT*, the new node and all associated compute is discarded. However, such a path may contain valid intermediary configurations. We propose to opportunistically add such configurations to the tree as if they were the new node—reusing spent compute and further progressing the tree towards the goal. This opportunistic approach forgoes the need for η by allowing the planner to attempt to connect the entire displacement from the current tree to the newly sampled node. If the current sample is unreachable from the nearest node in the existing tree, the planner opportunistically adds a collision-free node from earlier in the path. This enables the planner to grow the tree as much as the environment allows without incurring any additional cost to project or otherwise modify the old-sample/step-size parameter. The proposed strategy can be viewed as an extreme version of the connect heuristic that is leveraged every addition to the tree, which instead of incrementally stepping towards the sampled node takes the entire jump all at once. This opportunistic behavior eliminates the overhead incurred when adding incremental nodes through connect while still covering the space.

III. Opportunistic RRT*

The opportunistic approach presented in this paper, ORRT*, is a staged approach that conducts search in multiple phases. The first phase focuses on quickly finding a viable path from start to goal via a bidirectional search. After finding an initial path, a second phase collapses the bidirectional tree to a single tree, at which point informed sampling strategies are leveraged to efficiently improve the solution.

The first phase is performed through a combination of several opportunistic strategies: (1) efficiently reusing compute from collision checking, (2) stochastically "jumping" up to the entire distance towards the sampled node rather than stepping via a steering parameter η, and (3) greedily checking "one-shot" solutions similar to Zhang *et al.* [15].

The key insight behind (1) is that when connecting to a new node, expensive collision checks are made to verify that the sampled node is reachable. In standard RRT approaches, the sampled target node is rejected if a collision is found during this collision checking step. ORRT* rather treats the last viable configuration β steps away from the configuration found to be in collision as the actual target, effectively making use of the spent compute up to that point. This node is then added to the tree as usual and search is continued.

Provided (1), η (which restricts jumps taken by RRT to reduce wasted compute) can be eliminated. Instead of stepping at most η, ORRT* stochastically jumps up to the entire distance toward a sampled goal. These large jumps will not lead to wasted compute as before, as opportunistic nodes will be introduced if collisions are detected. This is achieved by generating a path[1] at a discretization appropriate for collision checking. The path is iterated from start to end, checking each configuration λ_i along the path for collisions. If a collision is found, we return a collision-free configuration β steps back, $\lambda_{i-\beta}$, from the in-collision λ_i. If the number of poses between the start of the interpolated path and the obstructed node is less than β, then we abandon the node and sample a new point. The elimination of η has surprising impacts on practical use of our approach—as

[1] An interpolation between the start and end configurations.

extensive η tuning for different environments/systems is no longer required.

As an additional heuristic to opportunistically improve planning performance, we attempt to directly attach all newly connected nodes to the goal in a single jump ((3) above). This one-shot solution can reduce planning time considerably by rapidly covering the problem state space. However, this behavior can exacerbate an issue introduced by opportunistic nodes: they can create "clusters" of opportunistic nodes around obstacles. These clusters contain nodes that are extremely close in configuration space and add little new information. Furthermore, these clusters slow down the rewiring performance by introducing very dense and similar neighbors that need to be iterated over for optimality. In an attempt to alleviate the impact of these clusters, we introduce an additional heuristic in which we do not allow an opportunistic node to be added to the tree if its parent is itself an opportunistic node[2].

After finding an initial path, the second phase of the approach leverages informed sampling to improve the current solution. Similar to Mashayekhi *et al.* [7], we collapse the forward and reverse trees to leverage the informed sampling technique of Gammel *et al.* to refine the initial solution and prune the search tree as tighter constraints are found [3]. In this study, we conducted experiments using systems with and without nonholonomic constraints. As such, we cannot arbitrarily invert the goal-rooted tree entirely while respecting tree constraints. To alleviate this, only the nodes along the found path are added to the forward tree to perform informed sampling with. This more restrictive tree-collapse approach was used in all evaluations. The modifications required for Opportunistic RRT* are restricted to the Extend* algorithm and are shown in blue in Algorithm 1.

A. Completeness

Previous work has shown that the RRT algorithm is probabilistically complete [17], with an exponential rate of decay for the probability of failure [19]. Because Opportunistic-RRT* adds all of the usual RRT nodes, in addition to other nodes which assist in covering the free configuration space, the approach inherits the probabilistic completeness guarantees from standard RRT methods.

B. Optimality

The work of Karaman and Frazzoli demonstrated that RRT* almost-surely converges to the optimal solution in the limit [2]. In that proof, an arbitrary η was used and the proof was shown to hold for any $\eta > 0$. In the proposed approach η is eliminated. With respect to the original proof, this effectively turns η into a dynamic parameter equal to the true distance between the sampled node and the nearest neighbor in the tree. As such, eliminating η and instead using the true distance has no impact on optimality at the limit, as rewiring is performed employing a radii that satisfies the bound presented in Karaman and Frazzoli [2].

[2]All opportunistic approaches employ this heuristic as it was found to improve plan times.

Algorithm 1: Extend*($G=(V,E),x,X_{soln}$)

```
/* Black text only for Extend* [2], add red text for
   informed sampling [3], add blue text for
   opportunistic approach presented in this paper.
   */
x_nearest ← Nearest(G,x)
~~x_new ← Steer(x_nearest,x)~~
x_new ← OppCollisionCheck(x_new,x_nearest)
if x_new ≠ x_nearest then
    V ← V ⋃ {x_new}
    x_min ← x_nearest
    X_near ← Near(G,x_new,r_RRT*)
    c_min ← Cost(x_nearest,G) + Cost(x_nearest,x_new)
    for each x_near ∈ X_near \ x_nearest do
        near_cost ← Cost(x_near,G) + Cost(x_near,x_new)
        if isCollisionFree(x_near,x_new) & near_cost < c_min then
            x_min ← x_near
            c_min ← near_cost
    E ← E ⋃ {x_min,x_new}
    for each x_near ∈ X_near \ x_min do
        near_cost ← Cost(x_new,G) + Cost(x_new,x_near)
        if isCollisionFree(x_new,x_near) & near_cost < Cost(x_near,G) then
            x_parent ← Parent(x_near,G)
            E ← E \ {x_parent,x_near}
            E ← E ⋂ {x_new,x_near}
    if x_new = x then return Reached
    else return Progress
    if InGoalRegion(x_new) then
        X_soln ← X_soln ∪ {x_new}
return Collision
```

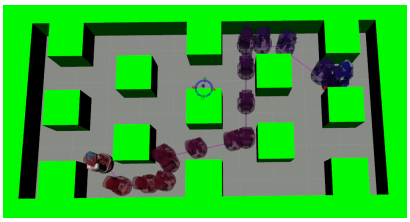
Algorithm 2: OppCollisionCheck($x_{new},x_{near},\beta=10$)

```
X_path ← Interpolate(x_new,x_near)
for i ← 0, to |X_path| do
    if nodeInCollision(X_path[i]) then
        if i > β then return X_path[i−β]
        else return x_near
return x_new
```

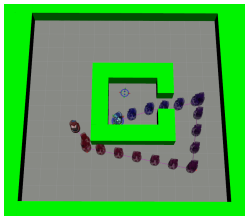
IV. Evaluation Settings

Four Opportunistic-RRT variants ORRT*, ORRTO*, ORRT, and ORRTO were evaluated. The four Opportunistic-RRT variants are differentiated by including the one-shot heuristic (trailing O) and if rewiring is used during the initial phase of the search (includes "*" in the abbreviation). Two rounds of evaluations were performed. The first evaluated the first path found by each approach compared to baseline RRT and RRT* approaches. The second round investigated the performance impact of additional sampling on each approach after initial paths are found. Hybrid (-H) variants collapse the initial bidirectional tree to a single tree and refine the path until an additional 100 nodes are added via Informed-RRT*. A standard bidirectional RRT* that adds an extra 100 uninformed nodes to the tree is also evaluated (RRT*-E). For each baseline $\eta = 3.0$, a fairly large value that favors quickly finding paths over optimality. No post-planning shortcuts were performed.

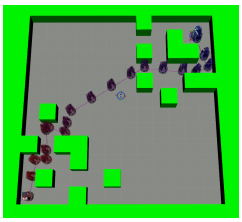
Evaluations were performed on two robot platforms and five environments shown in Fig. 2. A Zebra Fetch mobile manipulator is used to assess planar ($\mathbb{R}^3$) navigation planning performance in three environments. The first, Nav-A, is 12 m x 6 m (Fig. 2(a)) and contains regularly spaced checkerboard-like obstacles to stress the planner's ability to expand through clutter. The remaining environments (Fig. 2(b) and Fig. 2(c)) are 12 m x 12 m. Nav-B is a replication of the popular OMPL



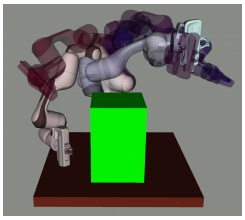
(a) Nav-A (Checkerboard)



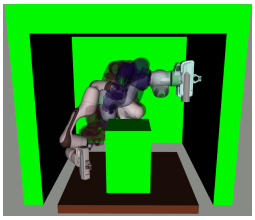
(b) Nav-B (Bug Trap)



(c) Nav-C (Cluttered Bug Trap)



(d) Open Panda (OP)



(e) Enclosed Panda (EP)

Fig. 2. Evaluation environments with arbitrary representative paths.

"bug trap" environment [20], which captures the planner's ability to "escape" difficult starting/goal configurations in open environments. Nav-C is designed to strike a balance between the clutter of the checkerboard environment and the "escape" difficultly of bug trap. The second platform is a 7-DOF ($\mathbb{R}^7$) Franka Emika Panda manipulator evaluated in a postural control task in two environments, one mildly and one highly constrained, shown in Fig. 2(d) and 2(e).

Planners were evaluated for their ability to find initial solutions, refine these solutions, and their sample efficiency in terms of the number of added and rejected nodes (attempted nodes). Each approach was run a total of 100 times in each setting in the same operating environment[3] and share the same code, allowing comparison of relative plan times. For the manipulators, a linear interpolation was used to generate paths in search. For the Fetch, a *Turn-Go-Turn* interpolation scheme was used. For each approach, the final geometric paths are parameterized using TOPP-RA [21].

V. Experimental Results

Planners were evaluated with respect to plan time and cost (total displacement in configuration space). To contrast the performance, two comparative measures were used: % S.P., which refers to the percentage of the time required by each approach with respect to the method that generated the (S)hortest (P)ath and P.D., the (P)ath (D)egradation of a specific approach as a multiple of the shortest path generated. Lastly, the sample efficiency of each approach was measured by tracking the total number of samples, both rejected and added to the tree, that are drawn during planning.

[3]Experiments where run in Ubuntu 20.04, ROS Noetic, on an Intel i9-12900 CPU with 64 GB of RAM.

A. Mobile Robot

The results of the planning experiments in the three navigation domains can be seen in Tables I and II for plan times and costs, respectively. The first six rows are the results from finding an initial path, while the remaining seven are additional sampling results.

For the first-path experiments, the proposed opportunistic planners significantly outperformed the baseline approaches in terms of plan time. The non-rewiring approaches performed the best, being on average 70.33% faster than RRT*, and 38.07% faster than the RRT baseline. The opportunistic variants with rewiring were on average 35.34% faster than RRT*. However, the plan time improvements seen in opportunistic methods come at the cost of path quality; being on average 1.28 and 1.55 times the length of RRT* generated paths for the rewiring and non-rewiring variants, respectively. Performance benefits from opportunistic nodes decrease as the environment becomes less cluttered, as can be seen by the performance of the opportunistic methods in the bug trap environment.

In the additional sampling experiments, we see similar results, where the opportunistic approaches generally outperform their baseline counterparts with respect to plan time in the more cluttered environments (by 62.00% for the rewiring methods and 53.72% for others), with performance degrading in bug trap to be on par with or slightly worse than the baselines (106.37% and 101.28%, respectively). These results indicate that opportunistic nodes are generally not as helpful in escaping narrow passages as they are in expanding through cluttered environments. This limitation is highlighted in bug trap because there is less opportunity to make further progress from a previously placed opportunistic node. Furthermore, because obstacles surround the goal, large clusters of nodes accumulate along the walls, resulting in more time being spent during rewiring. This issue is further exacerbated with one-shot approaches, which essentially add a node to these clustered sets whenever possible.

This analysis is further supported by the performance of opportunistic approaches in the third environment. This environment similarly walls-off the start and goal positions, but with clutter. The opportunistic approaches handily outperform their corresponding baselines, suggesting that the cluttered nature of these obstacles allows opportunistic nodes in the tree to jump around obstacles and quickly escape constrained configurations. The path length performance in the additional sampling settings see similar results to that of the first-path experiments, where the opportunistic approaches are generally longer than the baselines, but the gap between them narrows to 1.17 and 1.29 times the length of the shortest path for rewiring and non-rewiring approaches, respectively.

The impact of the environment on the performance of opportunistic approaches can be further seen in the average

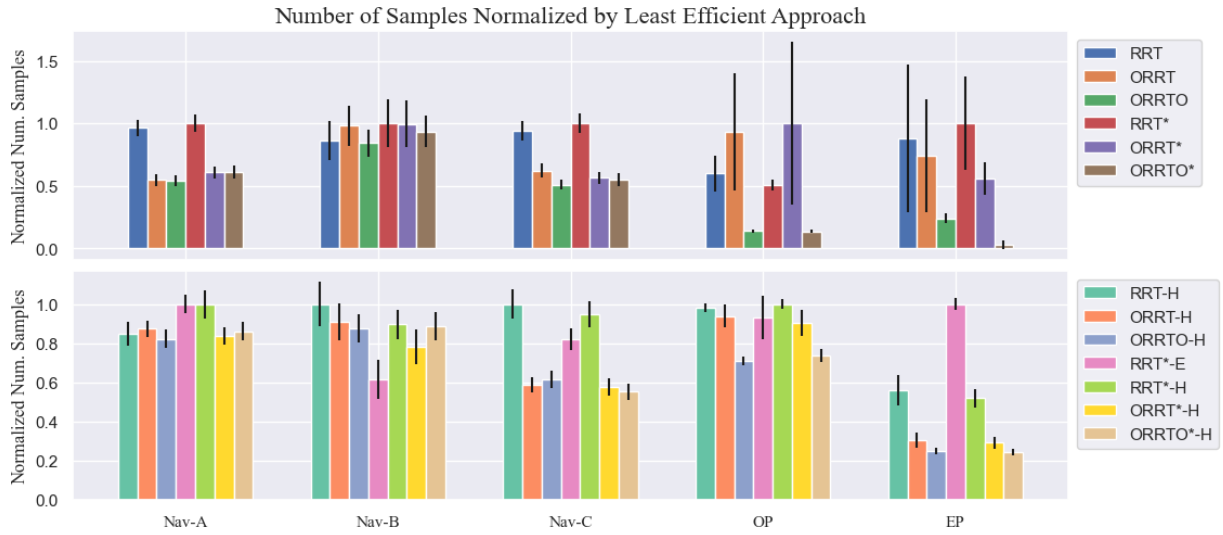


Fig. 3. Summary of the sample efficiency of each approach across the presented evaluation settings. In each environment, the total number of sampled configurations has been normalized by the least efficient approach (largest number of samples) in each setting. A lower score indicates that fewer samples were required to find a solution.

TABLE I

MEAN NAVIGATION ($\mathbb{R}^3$) PLAN TIMES (S), 100 RUNS WITH 95% CI

Approach	Nav-A	% S.P.	Nav-B	% S.P.	Nav-C	% S.P.
RRT	7.80 ± 0.50	0.68	4.07 ± 0.60	0.44	16.49 ± 1.20	0.44
ORRT	**3.09 ± 0.24**	0.27	**3.55 ± 0.47**	0.36	8.21 ± 0.73	0.22
ORRTO	3.19 ± 0.21	0.28	4.28 ± 0.48	0.43	**8.06 ± 0.89**	0.22
RRT*	11.36 ± 0.98	-	9.79 ± 2.49	-	37.12 ± 5.09	-
ORRT*	6.09 ± 0.67	0.53	8.67 ± 2.27	0.89	13.05 ± 2.11	0.35
ORRTO*	5.86 ± 0.63	0.51	7.40 ± 1.18	0.76	15.54 ± 2.22	0.41
RRT-H	29.83 ± 1.17	1.02	15.21 ± 1.28	0.82	28.56 ± 1.61	0.64
ORRT-H	**16.77 ± 0.60**	0.57	**14.69 ± 0.93**	0.79	**18.37 ± 0.75**	0.41
ORRTO-H	17.90 ± 0.56	0.61	16.12 ± 1.01	0.86	20.76 ± 1.34	0.47
RRT*-E	24.86 ± 1.36	0.85	18.65 ± 3.41	1.003	44.17 ± 5.26	-
RRT*-H	29.20 ± 1.37	-	18.59 ± 1.99	-	42.97 ± 5.14	0.97
ORRT*-H	19.56 ± 0.93	0.67	18.94 ± 2.91	1.02	26.53 ± 2.91	0.60
ORRTO*-H	18.11 ± 0.78	0.62	20.35 ± 1.83	1.10	25.96 ± 2.58	0.59

TABLE II

MEAN NAVIGATION ($\mathbb{R}^3$) SOLUTION COST, 100 RUNS WITH 95% CI

Approach	Nav-A	P.D.	Nav-B	P.D.	Nav-C	P.D.
RRT	31.21 ± 1.63	1.26	35.33 ± 1.63	1.19	41.91 ± 2.06	1.32
ORRT	38.64 ± 1.62	1.56	41.27 ± 2.06	1.39	51.17 ± 2.21	1.61
ORRTO	40.33 ± 2.01	1.62	44.45 ± 2.07	1.50	51.87 ± 1.88	1.63
RRT*	**24.83 ± 1.19**	-	**29.65 ± 1.04**	-	**31.73 ± 1.03**	-
ORRT*	30.73 ± 1.52	1.24	35.26 ± 1.46	1.19	39.59 ± 1.85	1.25
ORRTO*	32.15 ± 1.28	1.35	38.38 ± 1.63	1.29	42.91 ± 2.01	1.35
RRT-H	23.22 ± 1.09	1.14	30.54 ± 1.29	1.11	34.26 ± 1.32	1.12
ORRT-H	26.36 ± 1.25	1.30	34.81 ± 1.55	1.26	37.35 ± 1.58	1.22
ORRTO-H	27.17 ± 1.50	1.34	34.54 ± 1.48	1.25	42.63 ± 2.12	1.39
RRT*-E	20.48 ± 0.78	1.01	28.49 ± 0.99	1.03	**30.58 ± 0.86**	-
RRT*-H	**20.28 ± 0.93**	-	**27.62 ± 1.14**	-	30.73 ± 1.04	1.00
ORRT*-H	24.30 ± 0.93	1.20	31.11 ± 1.15	1.13	34.55 ± 1.18	1.13
ORRTO*-H	25.11 ± 1.18	1.24	32.87 ± 1.61	1.19	34.32 ± 1.36	1.12

number of samples drawn by each approach[4]. Fig. 3 displays the average number of samples per approach normalized by the largest number of samples in each setting. In the two cluttered environments, the opportunistic approaches require roughly half as many samples as the least efficient RRT* baseline approach in the first-path experiments and on average 28.5% fewer nodes in the additional sampling experiments. Notably, in the bug trap experiment, the disparity in the number of samples drawn closes significantly.

[4]Approaches sampled unlimited times to reach 100 additional nodes.

B. Manipulator Robot

Tables III and IV display the plan time and cost results from the postural control for a manipulator robot ($\mathbb{R}^7$) experiments. The ability of the one-shot heuristic to cut through the configuration space can clearly be seen in the plan time and cost results for the first-path experiments. The opportunistic approaches leveraging the one-shot heuristic were on average an order of magnitude faster than the other approaches and generated shorter paths than its counterparts in the "Open Panda" environment. However, we see that the plan times of the purely opportunistic approaches were on average 95.00% and 26.08% longer (for rewiring and non-rewiring, respectively) than their corresponding baselines. In the "Enclosed Panda" experiments, we see that the opportunistic approaches outperform their baselines by 34.09% and 13.51% respectively, while the one-shot enabled approaches perform the best, being on average 25.00% and 67.57% faster than the baselines. The path lengths generated by the opportunistic approaches were larger than those generated by the baselines, with the non-one-shot approaches performing the worst.

In the additional sampling experiments, opportunistic approaches take roughly twice as long as the baselines while generating significantly shorter paths. Further, we still see the same disparity between the performance of the pure-opportunistic planners and the one-shot planners in the open-domain in terms of path lengths. It is particularly noteworthy how much the paths are improved during additional sampling by the opportunistic approaches. These improvements yield paths much shorter than the baselines, even though the initial paths found by the baseline approaches were on average initially shorter. This suggests that opportunistic approaches are able to leverage opportunistic nodes introduced into the *useful* workspace of the robot due to self-collisions during additional sampling and are able to take additional shortcuts via RRT*-rewiring, resulting in much shorter paths overall.

TABLE III

MEAN MANIPULATOR ($\mathbb{R}^7$) PLAN TIMES (S), 100 RUNS WITH 95% CI

Approach	Open Panda	% S.P.	Enclosed Panda	% S.P.
RRT	0.23 ± 0.05	-	0.37 ± 0.20	3.08
ORRT	0.29 ± 0.09	1.26	0.32 ± 0.15	2.67
ORRTO	**0.06 ± 0.01**	0.26	0.12 ± 0.02	-
RRT*	0.20 ± 0.02	0.87	0.44 ± 0.14	3.67
ORRT*	0.39 ± 0.19	1.70	0.29 ± 0.06	2.42
ORRTO*	0.07 ± 0.004	0.30	**0.11 ± 0.01**	0.92
RRT-H	1.10 ± 0.11	0.59	**2.03 ± 0.18**	0.62
ORRT-H	1.95 ± 0.13	1.04	2.84 ± 0.23	0.86
ORRTO-H	1.86 ± 0.14	0.99	2.93 ± 0.25	0.89
RRT*-E	**0.68 ± 0.06**	0.36	2.17 ± 0.06	0.66
RRT*-H	1.05 ± 0.10	0.56	2.13 ± 0.17	0.65
ORRT*-H	1.94 ± 0.11	1.04	3.29 ± 0.27	-
ORRTO*-H	1.87 ± 0.12	-	3.12 ± 0.26	0.95

TABLE IV

MEAN MANIPULATOR ($\mathbb{R}^7$) SOLUTION COST, 100 RUNS WITH 95% CI

Approach	Open Panda	P.D.	Enclosed Panda	P.D.
RRT	**16.79 ± 0.83**	-	15.72 ± 0.87	1.001
ORRT	23.49 ± 1.25	1.40	17.42 ± 1.12	1.12
ORRTO	18.96 ± 0.67	1.13	**15.58 ± 0.88**	-
RRT*	16.81 ± 0.74	1.001	15.73 ± 0.81	1.01
ORRT*	23.96 ± 1.32	1.43	16.81 ± 1.02	1.08
ORRTO*	19.47 ± 0.64	1.16	15.84 ± 0.81	1.02
RRT-H	16.45 ± 0.83	1.38	16.08 ± 0.85	1.29
ORRT-H	14.58 ± 0.69	1.23	13.01 ± 0.75	1.04
ORRTO-H	11.90 ± 0.53	1.001	13.09 ± 0.68	1.04
RRT*-E	17.36 ± 0.98	1.46	16.21 ± 0.82	1.30
RRT*-H	17.09 ± 0.82	1.43	15.87 ± 0.75	1.29
ORRT*-H	13.81 ± 0.73	1.16	**12.50 ± 0.61**	-
ORRTO*-H	**11.89 ± 0.52**	-	12.72 ± 0.70	1.02

The performance degradation experienced by the opportunistic approaches in the open-manipulation experiment compared to the fenced-in experiment is particularly notable. In the enclosed panda experiments, we see a significant drop in the number of samples drawn compared to the open panda experiments. This is due to the fact that the environmental clutter begins to constrain the self-collision induced clutter from the system itself. Most significant of which is preventing the self-collisions that occur behind, to the sides, and above the robot arm, configurations which are entirely blocked by the fence in the enclosed environment. The fence is absent in the open experiment, so any sampled configurations that lead to self-collision in these regions will result in an opportunistic node being added to the tree. Furthermore, these opportunistic nodes are extremely unlikely to contribute to the final solution (initially or through additional sampling). Despite this, the opportunistic approaches will expend the full compute/time required to add these nodes to the tree, while the baseline approaches will simply abandon them. We believe this to be one of the primary contributing factors to the relatively poor timing/sample efficiency performance exhibited by the opportunistic approaches in the open experiment.

The sample efficiency and timing performance demonstrated by the one-shot enabled approaches in the open-panda, first-path manipulation-setting in comparison to enclosed panda and navigation domains suggest that the one-shot heuristic favors environments and robot configurations where the primary source of clutter is self-collisions. Further, when comparing to the navigation domains, where no self-collisions are possible, we see a significantly larger disparity in number of drawn samples between the one-shot and purely opportunistic approaches. This highlights the differences in performance from the primary source of collision being the result of the environment as opposed to self-collisions. Moreover, in the additional sampling experiments, we see that the opportunistic approaches are more sample-efficient than the baselines. This supports the assertion that opportunistic nodes induced by self-collisions in the usable workspace enable a greater ability for subsequent informed samples to improve the existing path.

VI. CONCLUSIONS AND FUTURE WORK

The opportunistic RRT* approach introduced in this work shows many promising characteristics and warrants further investigation. A key area of future work will be investigating how the proposed opportunistic behaviors perform with other sampling strategies, including: combining lazy sampling strategies with opportunistic behavior; exploring opportunism with respect to strategies that respect dynamics [22]; and using opportunistic nodes to inform and shape future samples similar to RRV [9] to grow samples orthogonal to detected obstacles. Anytime capabilities inherited from RRT* [23] also warrants investigation. The relative speed of the baseline opportunistic approaches, and the resulting improvements from additional sampling, indicate that anytime opportunistic behavior may lead to practical improvements in overall planning and execution times.

In our experiments collisions directly impacted opportunistic behavior. In cluttered environments, the opportunistic approach allows the tree to jump around obstacles and quickly escape constrained configurations. However, the presence of self-collisions directly impacts the ability of the proposed algorithm. Our collision detection module did not differentiate between configurations that resulted in self-collisions versus configurations that resulted in environmental collisions. The opportunistic behavior allows us to quickly consume the state space when only environmental collisions are considered. However, the presence of self-collisions leads to increased work in creating opportunistic nodes that do not necessarily help explore the configuration space of the problem. This leads to a degradation in planning times. Despite this, the nodes that are introduced from self-collisions may help generate shorter paths overall by giving the planner more maneuverability in the growing search tree, allowing greater opportunities to find shorter paths. Exploring these implications and the management of collision types is a promising area of future work.

ACKNOWLEDGEMENT

The authors would like to thank Khoshrav Doctor and Emily Pruc for feedback on the manuscript and colleagues at TRACLabs who supported development, particularly Kenji Brameld for support implementing interpolation schemes.

REFERENCES

[1] S. M. LaValle *et al.*, "Rapidly-exploring Random Trees: A new tool for path planning," 1998.

[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[3] 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," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2997–3004.

[4] O. Salzman and D. Halperin, "Asymptotically near-optimal RRT for fast, high-quality motion planning," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 473–483, 2016.

[5] M. P. Strub and J. D. Gammell, "Adaptively informed trees (AIT): Fast asymptotically optimal path planning through adaptive heuristics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3191–3198.

[6] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Informed sampling for asymptotically optimal path planning," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 966–984, 2018.

[7] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi, and I. Ahmedy, "Hybrid RRT: A semi-dual-tree RRT-based motion planner," *IEEE Access*, vol. 8, pp. 18 658–18 668, 2020.

[8] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2951–2957.

[9] A. Tahirovic and M. Ferizbegovic, "Rapidly-exploring random vines (RRV) for motion planning in configuration spaces with narrow passages," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7055–7062.

[10] B. Li and B. Chen, "An adaptive rapidly-exploring random tree," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 2, pp. 283–294, 2021.

[11] C. Wang and M. Q.-H. Meng, "Variant step size RRT: An efficient path planner for uav in complex environments," in *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, 2016, pp. 555–560.

[12] M. McCourt, C. T. Ton, S. S. Mehta, and J. W. Curtis, "Adaptive step-length RRT algorithm for improved coverage," in *AIAA Guidance, Navigation, and Control Conference*, 2016, p. 0638.

[13] B. An, J. Kim, and F. C. Park, "An adaptive stepsize RRT planning algorithm for open-chain robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 312–319, 2017.

[14] B. Liu, W. Feng, T. Li, C. Hu, and J. Zhang, "A variable-step RRT* path planning algorithm for quadrotors in below-canopy," *IEEE Access*, vol. 8, pp. 62 980–62 989, 2020.

[15] Y. Zhang, R. Wang, C. Song, and J. Xu, "An improved dynamic step size RRT algorithm in complex environments," in *2021 33rd Chinese Control and Decision Conference (CCDC)*. IEEE, 2021, pp. 3835–3840.

[16] S. Al-Ansarry, S. Al-Darraji, A. Shareef, D. G. Honi, and F. Fallucchi, "Bi-directional adaptive probabilistic method with a triangular segmented interpolation for robot path planning in complex dynamic-environments," *IEEE Access*, 2023.

[17] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[18] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann, "RRT*-Connect: Faster, asymptotically optimal motion planning," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2015, pp. 1670–1677.

[19] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of guidance, control, and dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[20] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.

[21] H. Pham and Q.-C. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 645–659, 2018.

[22] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2537–2542.

[23] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1478–1483.