

Revisiting Replanning from Scratch: Real-Time Incremental Planning with Fast Almost-Surely Asymptotically Optimal Planners

Mitchell E. C. Sabbadini¹, Andrew H. Liu², Joseph Ruan², Tyler S. Wilson¹,
Zachary Kingston², and Jonathan D. Gammell¹

Abstract—Robots operating in changing environments either predict obstacle changes and/or plan quickly enough to react to them. Predictive approaches require a strong prior about the position and motion of obstacles. Reactive approaches require no assumptions about their environment but must replan quickly and find high-quality paths to navigate effectively.

Reactive approaches often reuse information between queries to reduce planning cost. These techniques are conceptually sound but updating dense planning graphs when information changes can be computationally prohibitive. It can also require significant effort to detect the changes in some applications.

This paper revisits the long-held assumption that reactive replanning requires updating existing plans. It shows that the incremental planning problem can alternatively be solved more efficiently as a series of independent problems using fast almost-surely asymptotically optimal (ASAO) planning algorithms. These ASAO algorithms quickly find an initial solution and converge towards an optimal solution which allows them to find consistent global plans in the presence of changing obstacles without requiring explicit plan reuse. This is demonstrated with simulated experiments where Effort Informed Trees (EIT*) finds shorter median solution paths than the tested reactive planning algorithms and is further validated using Asymptotically Optimal RRT-Connect (AORRTC) on a real-world planning problem on a robot arm.

I. INTRODUCTION

A key application of motion planning algorithms in robotics is to find paths in environments with unknown or dynamic obstacles. Most planners use some combination of predictive or reactive approaches to find feasible or optimal paths quickly and efficiently [1]. Predictive approaches attempt to predict the movement of obstacles in the environment and generate plans around these predictions. Reactive approaches make exact plans using the known information and quickly modify them when changes are detected.

Predictive planners leverage *a priori* knowledge of their likely environments to improve their planning efficiency and path quality. When this includes the trajectories of dynamic obstacles, the planner can find plans that guarantee it does not collide with obstacles [1]. Planners may also inform their paths by avoiding sections of the environment that are likely to contain an obstacle [2] or ensuring their paths are valid within assumed obstacle parameters [3].

The quality of the paths found by predictive approaches depends heavily on the quality of the predictions. These can

¹ Estimation, Search, and Planning (ESP) Research Group, Queen's University, Kingston ON, Canada. {20ms116, 18tsw1, gammell}@queensu.ca.

² CoMMA Lab, Department of Computer Science, Purdue University, West Lafayette IN, USA {liu3458, ruan44, zkingston}@purdue.edu.

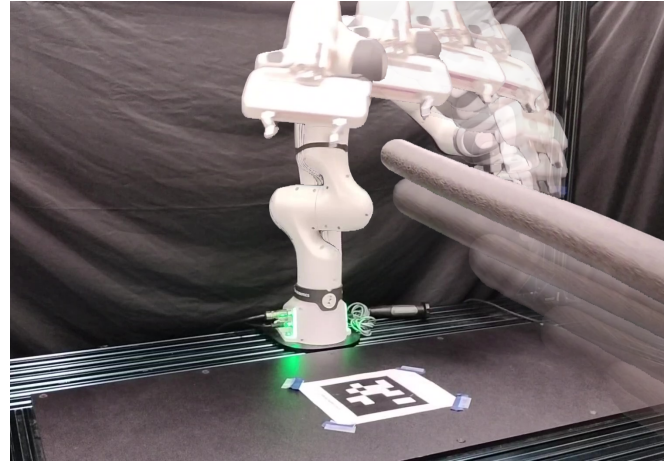


Fig. 1: A Franka Research 3 arm replanning around a dynamic obstacle by independently solving new planning problems each time the obstacle information changes. The VAMP [4] implementation of AORRTC [5] plans fast enough in a CAPT [6] obstacle map that it is able to replan without stopping the arm.

be poor in situations with limited preexisting data or highly complex dynamics. In such challenging environments, robots must pair their predictions with reactive planning.

Reactive approaches do not require assumptions about the position or motion of obstacles. These methods instead quickly generate intermediate plans when obstacle changes are detected. A successful reactive planner must efficiently generate high-quality intermediate paths while remaining consistent with the previously executed path.

Plan-reuse algorithms maintain consistency to previous plans by keeping the existing search and updating the parts affected by the change. This can allow them to provide formal guarantees on the optimality of their intermediate solutions (e.g., D* [7], LPA* [8], RRT^X [9]). They find better global paths than reactive approaches that focus solely on reaction times (e.g., bug algorithms [10]) but can be slower to react to changes in information.

Plan-reuse methods require the locations of obstacle changes in their search trees to efficiently maintain optimal intermediate paths. These locations may sometimes be available in the map representation but otherwise require manual evaluation to find. Changed edges can always be found by iterating through each edge in the search graph and confirming that it is still obstacle free. This increases the time it takes to update plans and may require a prohibitively large number of collision checks on large graphs.

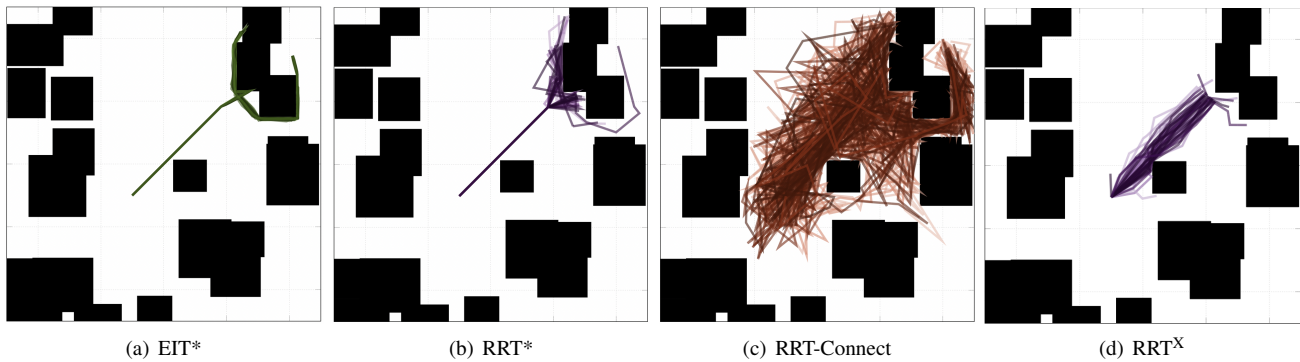


Fig. 2: A visualization of the paths travelled on 50 attempts of the incremental planning problem done by EIT*, RRT*, RRT-Connect and RRT^X with a planning budget of 0.1s. The hue of each path is determined by the order it was run in the experiment. The length of a planner’s global solution is the overall length of the path travelled. EIT* found the smallest median global solution length and had a 100% success rate. RRT* followed similar paths to EIT* due to its ASAO properties, when it was successful, but only succeeded 6% of the time. RRT-Connect also maintained a 100% success rate but had a longer median path length than EIT*. RRT^X had a 0% success rate due to the overhead of optimally rewiring its solution trees.

Recent almost-surely asymptotically optimal (ASAO) planners are able to quickly converge towards optimal solutions to complex problems. This provides an alternative for fast, globally optimal replanning without the complexity and computational overhead of plan reuse. Fast ASAO planners (e.g., Effort Informed Trees; EIT* [11], Asymptotically Optimal RRT-Connect; AORRTC [5]) can instead be run independently each time information changes to quickly find intermediate paths with optimality guarantees.

This challenges the historical assumption that fast incremental planning requires plan reuse [7] [12] [13]. This paper demonstrates that independent runs of EIT* outperform all other tested incremental planning algorithms on the simulated problems, including RRT^X, a planner specifically designed for efficient plan reuse. EIT* finds more and higher-quality solutions than all other planners across all tested planning budgets, successfully replanning as quickly as 50 ms in simulated experiments. It also evaluates this approach on a real robotic platform using AORRTC for moving-obstacle avoidance on a Franka Research 3 arm (Figure 1).

The rest of this paper is organized as follows. Section II presents a discussion of current and classic approaches to incremental planning. Section III presents further background and the independent incremental planning approach. Section IV presents the experimental details and Section V presents a discussion of the results. Finally, Section VI provides a conclusion and discusses future work.

II. LITERATURE REVIEW

Most replanning systems include a combination of predictive (Section II-A) and reactive (Section II-B) approaches.

A. Predictive Approaches

Predictive approaches leverage *a priori* information about their operational environments to inform their path plans but are susceptible to error in highly complex environments.

Space-Time RRT* (ST-RRT*) [1] uses ASAO planning to find paths through moving obstacles with optimality guarantees when their trajectories are known. ST-RRT* uses an ASAO planner in a space augmented with a time dimension

to model obstacle movements. This approach yields optimal plans through the moving obstacles but assumes perfect knowledge of their trajectories.

Velocity obstacle approaches [3] [14] assume that obstacles move up to a predetermined velocity and plan a trajectory to the goal avoiding any region that may be occupied by an obstacle under this constraint. Velocity constraints ensure the feasibility of solution paths but conservative bounds may restrict the quality of the solutions found and can create unsolvable problems.

Finite-horizon Model Predictive Control (MPC) can also be used to avoid obstacles. These approaches simultaneously react to obstacle changes and predict future trajectories by solving a finite-horizon optimal control problem defined by obstacle predictions. This replans quickly but provides no optimality guarantees on the intermediate paths if the problem horizon is shorter than the path to the goal [15].

B. Reactive Approaches

Reactive approaches do not require assumptions about the dynamics of their environments but must be able to quickly replan to maintain feasibility through the environment.

Bug algorithms [10] avoid expensive planning by following a deterministic policy. Basic bug algorithms drive a straight-line path to the goal when possible and follow the perimeter of any obstacles they encounter. Bug algorithms react quickly to new obstacles but provide no optimality guarantees and often execute highly suboptimal paths.

Following suboptimal intermediate paths may result in the robot unnecessarily changing the homotopy class of its path which can lead to a much longer global solution. Many existing approaches avoid this problem by providing guarantees of near-optimal intermediate paths.

Zelinsky [16] presents an algorithm that replans from scratch with A* once environment changes are detected. This approach guarantees resolution-optimal intermediate paths but suffers from slow intermediate path formulation.

D* [7, 17–19] extends the A* algorithm to replan when information changes. It improves the replanning speed by propagating local graph updates through the previous graph once

changes are detected. This is proven to find the resolution optimal solution given the information available at the time of planning but assumes it is given all graph changes and does not account for the computational cost of detecting them.

Lifelong Planning A* (LPA*) [8] also directly extends A* to incremental planning. It further improves replanning speed by only propagating updates through edges that are affected by graph changes. LPA* also assumes knowledge of which edges in the graph have changed and does not consider the computational cost of detecting changes.

Lifelong Generalized Lazy Search (L-GLS) [20] extends LPA* to take advantage of the efficiency of lazy search algorithms. This finds the resolution optimal solution faster than other graph-based incremental replanners by deferring expensive edge checks until necessary but does not avoid the computational costs of detecting changes.

Execution extended RRT (ERRT) [12] extends the Rapidly-exploring Random Trees (RRT) algorithm [21] to incremental planning. ERRT samples states from previous planning efforts to bias its search toward previous paths. This can reduce the replanning time if obstacle locations are similar between planning efforts but provides no guarantees about the quality of its intermediate solutions.

Dynamic RRT (DRRT) [13] also extends RRTs to incremental planning. DRRT reuses feasible branches of previous planning efforts to speed up intermediate path planning but it must validate all of its previous edges once obstacle changes are detected and makes no guarantees about the quality of its intermediate solutions.

RRT^X [9] extends RRT* [22] to apply ASAO sampling-based planning methods to incremental planning. RRT^X efficiently repairs its path as information changes and plans from the goal state to avoid unnecessary rerooting of the tree once the planner's location is updated. RRT^X does not consider the computational costs of detecting which edges in its solution tree are invalidated by obstacle changes.

Multiobjective Dynamic RRT* (MOD-RRT*) [23] uses a modified RRT* to find an initial path to the goal and then uses a rewiring method to quickly repair paths that collide with obstacles near the robot. This rewiring does not explicitly maintain path optimality and therefore provides no guarantees around intermediate path quality. MOD-RRT* avoids expensive rewiring cascades by only rewiring locally and not propagating costs through the tree but this reduces the quality of its global solutions.

C. Independent Replanning

Any single-query planner can be used for incremental planning by solving a new independent planning problem each time new obstacles are detected. This does not require specific knowledge of obstacle changes and therefore avoids the computational overhead of many replanning methods. When these problems are solved optimally, the resulting global path will avoid unnecessary changes of direction (i.e., homotopy class switches) but each independent problem must be solved quickly enough to react to obstacle changes.

Recent ASAO planners, such as EIT* [11] and AORRTC [5], quickly find high-quality solutions and their intermediate paths may be combined into high-quality global solutions without the extra computational cost of replanning.

III. METHODOLOGY

Incremental planning is an extension of optimal planning with imperfect information about the environment.

A. Optimal Planning

The optimal planning problem is defined similarly to [22]. Let $X \subseteq \mathbb{R}^n$ be the state space of the planning problem, $X_{\text{obs}} \subset X$ be the set of states in collision with obstacles, and $X_{\text{free}} = \text{closure}(X \setminus X_{\text{obs}})$ be the resulting set of permissible states. Let $x_{\text{start}} \in X_{\text{free}}$ be the initial state and $X_{\text{goal}} \subseteq X_{\text{free}}$ be the set of desired final states. Let $\sigma : [0, 1] \rightarrow X$ be a sequence of states (i.e., a path) and Σ be the set of all paths.

The optimal solution is the path, $\sigma^* \in \Sigma$, that minimizes a chosen cost function, $c : \Sigma \rightarrow \mathbb{R}_{\geq 0}$, while connecting the start, x_{start} , to any goal, $x_{\text{goal}} \in X_{\text{goal}}$, through free space,

$$\sigma^* = \arg \min_{\sigma \in \Sigma} \left\{ c(\sigma) \mid \sigma(0) = x_{\text{start}}, \sigma(1) \in X_{\text{goal}}, \right. \\ \left. \forall t \in [0, 1], \sigma(t) \in X_{\text{free}} \right\},$$

where $\mathbb{R}_{\geq 0}$ denotes the set of all real numbers greater than or equal to zero.

B. Incremental Planning

The incremental planning problem is an extension of the optimal planning problem where the locations of obstacles in the environment are not perfectly known and may change as the robot moves. Exact knowledge of obstacles is assumed to be limited to those detected in a specified distance of the robot, r_s , within a retrospective time horizon, τ .

Let $B(r_s, x) \subset X$ denote a sensing region (e.g., a ball) with maximum sensing range, $r_s \in \mathbb{R}_{\geq 0}$, and a center point, $x \in X$. The robot has exact knowledge of obstacles that are within the sensor range of its current position, i.e., $X_{\text{sensed}}(t) = X_{\text{obs}}(t) \cap B(r_s, x)$, where $X_{\text{obs}}(t)$ is the obstacles at time t . Let $X_{\text{sensed},i}$ be the set of all obstacles sensed by the robot and considered for planning at the i^{th} iteration, $X_{\text{sensed},i} = \bigcup_{t=t_i}^{t_j} X_{\text{sensed}}(t)$, where $t_i = t_j - \tau$ is the minimum time from which the planner still uses its previous obstacle measurements. The robot may also include a prior over obstacle motions in its set of obstacles.

The robot plans in the free space defined by the set of obstacles sensed within the retrospective time horizon, $X_{\text{free},i} = X \setminus X_{\text{sensed},i}$, and follows the resulting solution path, σ_i , until it reaches a state at which it determines it must replan from, x_{i+1} . This process repeats until the robot reaches the goal, X_{goal} .

Let $s_i \in [0, 1]$ denote the parameter at which the robot replans from when following its i^{th} intermediate solution path, i.e., $x_{i+1} = \sigma_i(s_i)$. Let $\sigma_{i,s_i} : [0, s_i] \rightarrow X$ denote the partial path travelled during the i^{th} planning iteration. The global solution path is defined as $\pi = \sigma_{1,s_1} \oplus \sigma_{2,s_2} \oplus \dots \oplus$

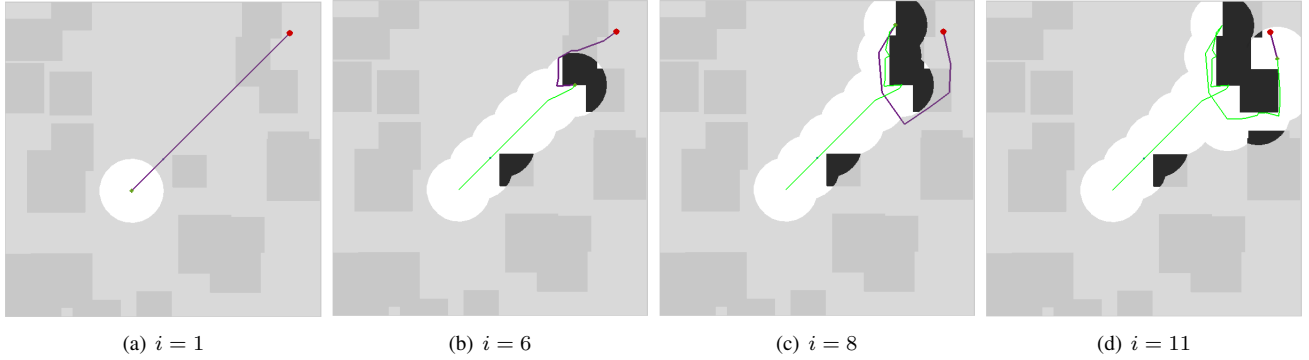


Fig. 3: A visualization of the intermediate paths found by EIT* with a sensor range of 0.1 and a planning budget of 0.1s. The gray area is unsensed and assumed to be free space by the planner. The dark green dot represents the starting point of the query, and the red dot represents the goal. The green path represents the path travelled so far. The number of circles gives a representation of the number of queries of the planner.

σ_{N,s_N} , where \oplus denotes path concatenation and $N \in \mathbb{Z}_{>0}$ is the number of planning cycles until the goal is reached. An incremental planner seeks to minimize the cost of its global solution, $c(\pi)$.

C. Independent Incremental Planning Approach

The independent incremental planning approach uses independent calls to ASAO planners to solve the incremental planning problem. The robot solves an independent optimal planning problem in the free space defined by the sensed obstacles at each planning iteration, $X_{\text{free},i} = X \setminus X_{\text{sensed},i}$, to find a path, σ_i . The robot follows the path until it arrives at a state at the edge of the sensed area or otherwise determines it needs to replan, x_{i+1} . The robot updates its obstacles using the sensor and then replans from the new position. The process of planning, following the solution path and sensing repeats until the robot reaches the goal.

Let σ_i^* be the shortest path in the known free space from the current state, x_i , to the goal. Let σ_{i,s_i}^* denote the subpath of σ_i^* that ends at the next sensing boundary. Following the optimal path with respect to the given information at each iteration, σ_{i,s_i}^* , minimizes the worst-case total path cost over all possible environments, given the current information.

The quality of a global solution to the incremental planning problem depends on the quality of its intermediate paths. If the planner finds sufficiently optimal intermediate solutions then the executed path will avoid oscillating between different homotopy classes (i.e., be consistent) without explicitly considering path consistency and result in a near optimal global solution. If the planner does not provide optimality guarantees then it may yield suboptimal intermediate paths that are inefficient, backtrack, and/or unnecessarily switch homotopy classes and generally reduce the quality of the global solution.

IV. EXPERIMENTS

The independent incremental planning approach was tested on both probabilistically complete (RRT-Connect [24] with and without smoothing) and ASAO planners (RRT* and EIT*) and was compared against a state-of-the-art planner designed for incremental replanning problems (RRT^X). Since the planning time of RRT^X is slow when it has to optimally

Algorithm 1: Independent Incremental Planning

Given: $X, x_0, x_{\text{goal}}, r_s, \text{planner}$

- 1 $X_{\text{sensed},0} \leftarrow X_{\text{obs}}(0) \cap B(r_s, x_0)$;
- 2 $X_{\text{free},0} \leftarrow X \setminus X_{\text{sensed},0}$;
- 3 $\pi \leftarrow \emptyset$; $x_i \leftarrow x_0$;
- 4 **repeat**
- 5 $\sigma_i \leftarrow \text{planner}(x_i, x_{\text{goal}}, X_{\text{free},i})$;
- 6 $\pi \leftarrow \pi \oplus \sigma_{i,s_i}$;
- 7 $x_j \leftarrow \sigma_i(s_i)$;
- 8 $c(\pi) \leftarrow c(\pi) + c(\sigma_{i,s_i})$;
- 9 $X_{\text{sensed},i} = \bigcup_{t=t_i}^{t_j} X_{\text{sensed}}(t)$;
- 10 $X_{\text{free},i} \leftarrow X \setminus X_{\text{sensed},i}$;
- 11 $x_i \leftarrow x_j$;
- 12 **until** $x_i = X_{\text{goal}}$;
- 13 **return** π

rewire dense solution trees, results are presented for both the full version of RRT^X and a version that stops after it finds an initial solution to provide best-case results for RRT^X with respect to initial solution time.

Each of the planners were tested on three types of simulated worlds using publicly available Open Motion Planning Library (OMPL) [25] implementations with the partial RRT^X implementation publicly available in OMPL extended to handle replanning. EIT* used a batch size of 100, radius factor of 1.001, repair factor of 1.2 and used the k -nearest implementation with pruning. RRT-based planners used a maximum edge length of 0.3 and single-tree RRT-based planners used a goal bias of 0.05. RRT^X was run without informed sampling and an epsilon of zero. RRT* was run with a rewire factor of 1.001.

The incremental planning problem was simulated as an extension of the Planner Development Tools (PDT) [26]. The simulation requires three parameters to describe the environment: a planning budget, t , a maximum sensor range, r_s , and a global problem. The environment the planner interacts with for each intermediate problem is referred to as the *incremental* environment. The simulation assumes that all sensed obstacles are stationary, i.e., $\tau = \infty$.

The incremental environment starts out empty with the same start, x_0 , and goal, X_{goal} , as the global environment.

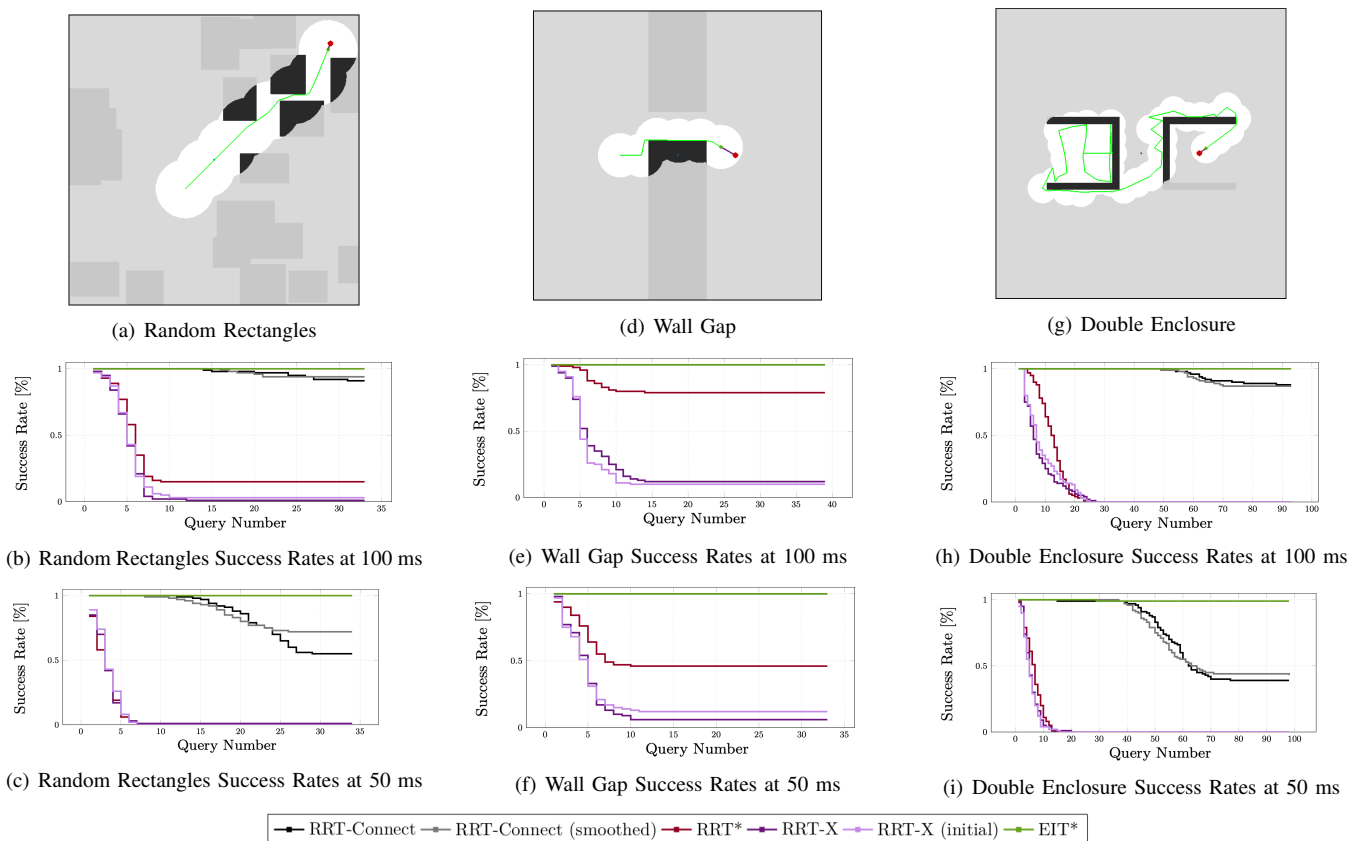


Fig. 4: The success rates of all planners across 100 runs on three of the simulated worlds. A visualization of the world as well as a solution found by EIT* is shown in (a), (d) and (g). Each of the planners was tested with a planning budget of 0.1s and 0.05s. The success rate at the n^{th} query is defined by the percentage of planners that are able to find a solution to the n^{th} intermediate problem within the planning budget. Success rates are carried forward, i.e., if a planner fails on a query it is also considered to fail on all subsequent queries. The independent ASAO approach with EIT* maintained the highest success rate across every world and every planning budget while finding the shortest global solution (Table I).

Only the obstacles within the sensor range from the start are added, and the planner attempts to find a path from the start to the goal that avoids these obstacles before the planning budget runs out. If the planner succeeds in finding a solution, the robot follows this path until it reaches a state at the end of the sensor radius, x_1 . Once the robot has moved, a circle centered at its current position with a radius equal to r_s , i.e., $B(r_s, x_1)$, is added to the sensed area. The next incremental problem is defined as a new optimal planning problem using x_1 as its starting point. The obstacles in this problem are the intersection of all obstacles in the global environment and the (now larger) sensed area. This process is repeated until the planner reaches the goal (Figure 3). This sequential planning problem under the free space assumption occurs when planning with limited sensing range and also for dynamic obstacles without any predictions.

The performance of RRT^X depends on the computational cost of detecting edges that are invalidated by newly discovered obstacles. This can be computationally inexpensive in map representations that provide this information directly or expensive when each existing edge must be checked each time new obstacle information is received. In order to provide best-case results for RRT^X, the computational cost of detecting invalidated edges for RRT^X was ignored in the experiments.

A. Simulated Experiments

The simulated worlds are two-dimensional path length minimization problems in a square environment, $X = [-1, 1]^2$. A sensor range of 0.1 was used on each Random Rectangles world, 0.075 on the Wall Gap world and 0.05 on the Double Enclosure world.

1) *Random Rectangles*: Each random world consists of 20 randomly placed rectangles with side lengths uniformly distributed between 0.1 and 0.2. The starting state is $(-0.1, -0.1)$, and the goal state is $(0.4, 0.4)$. Each world is checked for validity (i.e., a feasible path exists) before it is used. Figure 4a illustrates a representative world for which individual results are presented. The random worlds evaluate the capability of each planner to quickly navigate around many obstacles to reach the goal.

2) *Wall Gap*: The wall gap world (Figure 4d) has two rectangles as obstacles centered about the y-axis. There is a narrow rectangular gap between the two rectangles that the planner must navigate through. The wall gap tests the ability for planners to find and navigate through a narrow gap at the center of the world.

3) *Double Enclosure*: The double enclosure is two rectangular enclosures, each with one open end, containing the start and goal (Figure 4g). The planner initially sees an unobstructed path from start to goal but will discover it is

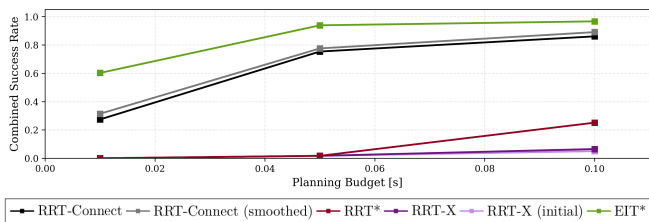


Fig. 5: The combined success rate of each planner on 100 different random worlds for planning budgets of 0.01s, 0.05s and 0.1s. EIT* had the highest total success rate at each planning budget and RRT^X was not able to consistently solve the problems in the available time.

blocked after travelling towards it. The planner must then retrace back towards the start to leave the first enclosure and then circumnavigate the second enclosure to reach the goal. The double enclosure tests the planner’s ability to adapt when initial plans are proven invalid by new information.

Three sets of experiments were run using the simulated worlds. Every planner was run 100 times on the Wall Gap problem, Double Enclosure problem and 40 different Random Rectangles problems with a planning budget of 0.1s to investigate the performance of each planner on a diverse set of problems. Every planner was run 100 more times on the Wall Gap problem, Double Enclosure problem and a representative Random Rectangles problem with a planning budget of 0.05s to investigate their performance on representative problems with a smaller budget. Each planner was also run 10 times with planning budgets of 0.01s, 0.05s and 0.1s on 100 different Random Rectangles problems for each planning budget to investigate the performance of each planner at different planning budgets (Figure 5).

B. Results

The per-query success rate, global solution time, global path length, and number of queries were measured for each simulated world. The combined success rate, median problem success and median rank of each planner’s median global path length, global solution time, and number of queries were also collected across the 40 random worlds. The combined success rate across all runs over the 100 random worlds was collected for each planning budget. A summary of the results from a planning budget of 0.1 seconds is presented in Table I. Only the success rates are shown for the 50 ms experiments (Figure 4) since EIT* was the only planner that solved more than 50% of all trials on all of the worlds. A plot of the total success rate over 100 random worlds at each planning budget is shown in Figure 5.

1) *Success Rates:* The success rate for each query is shown in Figure 4. The success rate at a given query is a measure of how often the planner is able to find a solution within the given planning budget at that query. The global success rate (i.e., the percentage of runs that the planner reached the goal) can be extracted from the success rate at the final query. EIT* had the highest global success rate on all simulated worlds at all evaluated planning budgets. None of the evaluated planners had 100% success on all problems because some randomly generated worlds require solutions to

pass through very narrow gaps to reach the goal. Increasing the planning budget further on these difficult problems would increase their success rate.

2) *Global Path Length:* The median global path length is shown in Table I. The global path length measures how far each planner travelled from the start to the goal. The global path length is the length of the portions of the intermediate paths followed to travel between the start and goal. EIT* had the shortest median global path length on all problems due to the high-quality intermediate paths it finds as a result of its ASAO properties.

3) *Global Solution Time & Queries:* The median global solution time and median number of queries are shown in Table I. The global solution time is the total time spent finding a solution to the incremental planning problem. The number of queries is the number of intermediate queries a planner took to reach the goal.

Nonoptimal planners, such as RRT-Connect, do not spend the whole planning budget and instead return as soon as they find an initial solution. Optimal planners, such as RRT* and RRT^X, first find an initial solution and then spend the rest of the planning budget searching for shorter paths. Other optimal planners, such as EIT*, have the capability to exit early if they detect that they have found the optimal solution. The number of queries is presented alongside the global solution time to give a fair comparison between planners.

EIT* had the smallest median number of queries on all problems. RRT-Connect had a smaller median global solution time than EIT* on the Wall Gap and Double Enclosure problems since it does not use the whole planning budget. EIT* maintained the smallest median global planning budget on the selected Random Rectangles problem despite spending the whole planning budget optimizing its paths due to the small number of queries it needs to solve the problem.

C. Real-World Planning

A VAMP [4] implementation of AORRTC was used as the incremental planner to navigate a seven DoF Franka Research 3 robot arm around moving obstacles. A collision-affording point tree (CAPT) [6] was used to efficiently map obstacles presented in different configurations and AORRTC planned a new solution around each of the different obstacles with no prior knowledge of their locations.

AORRTC successfully navigated past each obstacle configuration in real-time due to its ability to quickly find high-quality intermediate paths. The robotic arm experiment demonstrates the ability for fast ASAO planners to navigate successfully in high-dimensional environments with moving obstacles without modelling their motion.

V. DISCUSSION

The simulated experiments show that the incremental planning problem can be solved by independent planning and that when these planners find high-quality solutions quickly this independent incremental planning approach outperforms specifically designed incremental planners.

RRT-Connect was able to quickly find solutions to intermediate problems but had larger median global path

TABLE I: A summary of all planning results on the Random Rectangles, Wall Gap and Double Enclosure simulated worlds. Each result shows the percentage of successful runs, the median solution path length, the median solution time, and the median number of queries taken to reach the goal. Clopper-Pearson 95% confidence intervals are shown in brackets for single-environment success rates and nonparametric 95% confidence intervals for the median of all other metrics. The best result on each environment is shown in bold. Unsuccessful trials were considered to have infinite path length, solution time, and queries. The combined success rate, median success rate and the median rank of each planner’s median path length, median global solution time and median number of queries are shown for the 40 different Random Rectangles worlds.

Environment	Metric	Planner					
		RRT-Connect	RRT-Connect (smoothed)	RRT*	RRT ^X	RRT ^X (initial)	EIT*
Rand. Rect. (Example)	Succ. Rate	91% [83.6, 95.8]	94% [87.4, 97.8]	9% [8.6, 23.5]	1% [0.0, 5.4]	3% [0.6, 8.5]	100% [96.4, 100.0]
	Path Len.	4.78 [4.15, 5.34]	3.14 [2.93, 3.34]	∞	∞	∞	0.718 [0.717, 0.719]
	Sol. Time	0.366s [0.291, 0.388]	0.540s [0.451, 0.638]	∞	∞	∞	0.313s [0.313, 0.313]
	Queries	23 [21, 25]	18 [17, 19]	∞	∞	∞	7 [7, 7]
Wall Gap	Succ. Rate	100% [96.4, 100]	100% [96.4, 100]	79% [69.7, 86.5]	12% [6.4, 20.0]	10% [4.9, 17.6]	100% [96.4, 100]
	Path Len.	1.95 [1.65, 2.15]	1.72 [1.37, 1.84]	0.543 [0.522, 0.615]	∞	∞	0.389 [0.388, 0.390]
	Sol. Time	0.070s [0.054, 0.074]	0.073s [0.062, 0.087]	0.705s [0.608, 0.706]	∞	∞	0.399s [0.399, 0.399]
	Queries	15 [15, 18]	13 [13, 15]	7 [6, 7]	∞	∞	5 [5, 5]
Double Enclosure	Succ. Rate	88% [80.0, 93.6]	87% [78.8, 92.9]	0% [0.0, 3.6]	0% [0.0, 3.6]	0% [0.0, 3.6]	100% [96.4, 100]
	Path Len.	6.32 [5.90, 6.90]	6.05 [5.69, 6.52]	∞	∞	∞	2.01 [1.99, 2.02]
	Sol. Time	1.14s [1.09, 1.21]	1.88s [1.72, 2.10]	∞	∞	∞	2.50s [2.40, 2.50]
	Queries	70 [67, 72]	66 [63, 68]	∞	∞	∞	31 [31, 31]
Rand. Rect. (Agg.)	Total Succ.	95.2%	96.0%	26.2%	8.13%	8.70%	98.9%
	Med. Succ.	99.0% [99.0, 100]	100% [99.0, 100]	18.0% [11.0, 35.0]	4.0% [1.0, 8.0]	2.0% [1.0, 9.0]	100% [100, 100]
	Med. Rank	3/1/3	2/2/2	4/4/4	4/4/5	4/4/6	1/3/1

lengths than the ASAO planners. The lack of solution quality guarantees meant that RRT-Connect did not find consistent paths, both with and without smoothing. This resulted in paths that unnecessarily left the sensed area or changed homotopy classes and resulted in more planning queries than necessary. The smoothed variation of RRT-Connect outperformed unsmoothed RRT-Connect on path length and number of queries but had a longer median path length and larger number of queries than EIT* across all problems. Both variations of RRT-Connect had lower total success rates than EIT* on all sets of random worlds because their inconsistent intermediate solutions cause them to solve an unnecessarily large number of planning queries to reach the goal.

RRT^X failed to find a complete global solution within the incremental planning budget on more than 90% of its trials across all simulated worlds. This is because RRT^X spends computational effort to update its entire search tree each time obstacles change, even if parts of it are not necessary to solve an individual query. RRT^X has to rewire potentially very large trees when there are significant obstacle changes that affect many edges in its solution tree, which is computationally expensive.

The computational cost of this tree update also increases with the planning budget. A large planning budget means that more nodes will be inserted during each query and therefore need to be rewired and collision checked when updating the tree in future queries. This is an inherent trade-off of anytime-resolution plan-reuse algorithms, since the tree becomes more dense with more time between queries. This has been addressed in multiquery settings by rewinding the sample density between queries [27].

The computational cost of optimally rewiring dense solution trees is further demonstrated by the performance of

RRT^X when it is limited to only finding an initial solution. This truncated version of RRT^X was able to find solutions to more intermediate queries than the full RRT^X on most simulated experiments but failed to find a complete global solution within the incremental planning budget on more than 90% of the trials.

RRT* was able to find shorter median global path lengths than RRT-Connect and RRT^X but struggled to find initial solutions on small planning budgets. The ASAO properties of RRT* ensure that when it is given sufficient planning time it converges towards the optimal solution. This consistency between intermediate solutions ensures that RRT* only exits the sensed area when necessary but the required planning budget proved to be large. RRT* failed more than half the time on the Double Enclosure problems and all sets of Random Rectangles due to its slow initial solution time.

EIT* is able to find fast and near-optimal intermediate solutions since it is designed to quickly find an initial solution and then converge toward the optimal path. EIT* found shorter median solution paths than all other tested planning algorithms at all planning budgets. EIT* had the highest success rates on all sets of Random Rectangles problems at every planning budget and was the only planner to solve more than 50% of the problems on all experiments.

The real-world experiments demonstrate the applicability of the independent ASAO replanning strategy to other fast ASAO planning algorithms, dynamic obstacles, and real-world robotics problems. Suitably fast ASAO planners are able to replan each time obstacle changes are detected and achieve real-time dynamic planning, as demonstrated in the Franka arm experiment.

The simplicity of implementing the independent ASAO approach is also worth noting. Replanning from scratch simplifies many modeling assumptions in dynamic environments by not relying on a prior or attempting to reuse information between queries. An incremental planner that tries to reuse information from previous planning efforts would incur a performance overhead on long-horizon planning problems like the Franka arm experiment and would require careful tuning or sophisticated methods to control its solution-tree density. Many existing dynamic planners would require prior information about their obstacles like a maximum velocity or probabilities of different obstacle trajectories to navigate in the experiment. The only parameter tuning needed to run the independent ASAO replanning strategy is to use the largest budget that can be afforded and no problem-specific information or prior data is required, although it can be incorporated into the obstacle map if available.

VI. CONCLUSION AND FUTURE WORK

Traditional reactive planning literature focuses on trying to reuse information from previous queries in order to speed up planning time. This is often done by attempting to repair old plans when new obstacle information becomes available in order to maintain their path validity, consistency, and/or optimality. For this to be computationally tractable, many current algorithms assume that there is an efficient method to identify the locations of obstacle changes and propagate this information throughout their existing plans.

This paper shows that independent calls to ASAO planners can instead outperform these planners on incremental planning problems. We demonstrate that a suitably fast ASAO planner, such as EIT*, outperforms information-reuse algorithms, such as RRT^X, even when access to obstacle changes is free. It also shows that EIT* finds lower-cost solutions more quickly and with a higher success rate than independent runs of other ASAO algorithms, such as RRT*, and fast probabilistically complete algorithms like RRT-Connect.

The results of this paper demonstrate the capability of ASAO planners to replan in simulation as quickly as 50 ms and the applicability of this approach to a real-world robotic arm problem. This capability is exciting as it shows the ability to find near-optimal global solutions at close to the speed of control-level systems. In the future we would like to investigate using the independent ASAO approach on robots with obstacle predictions to find better solutions in environments with dynamic obstacles and on robots with kinodynamic or other constraints.

VII. ACKNOWLEDGEMENTS

Sneha Ramshanker did an early investigation for this work at the University of Oxford. This work was supported by a Charles Allan Thompson Award and the Natural Sciences and Engineering Research Council of Canada (NSERC) [RGPIN-2024-06637].

REFERENCES

- [1] F. Grothe, V. N. Hartmann, A. Orthey, and M. Toussaint, "ST-RRT*: Asymptotically-optimal bidirectional motion planning through space-time," in *ICRA*, 2022.
- [2] B. J. Luders, M. Kothari, and J. P. How, "Chance constrained RRT for probabilistic robustness to environmental uncertainty," in *AIAA Guidance, Navigation, and Control Conference*, 2010.
- [3] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *IJRR*, 1998.
- [4] W. Thomason, Z. Kingston, and L. E. Kavraki, "Motions in Microseconds via Vectorized Sampling-Based Planning," in *ICRA*, 2024.
- [5] T. S. Wilson, W. Thomason, Z. Kingston, and J. D. Gammell, "AORRTC: Almost-surely asymptotically optimal planning with RRT-Connect," *RA-L*, 2025.
- [6] C. Ramsey, Z. Kingston, W. Thomason, and L. E. Kavraki, "Collision-Affording Point Trees: SIMD-Amenable Nearest Neighbors for Fast Motion Planning with Pointclouds," in *RSS*, July 2024.
- [7] A. Stentz, "The D* algorithm for real-time planning of optimal traverses," Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-94-37, September 1994.
- [8] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artificial Intelligence*, vol. 155, no. 1–2, pp. 93–146, 2004.
- [9] M. Otte and E. Frazzoli, "RRT^X: Real-time motion planning/replanning for environments with unpredictable obstacles," in *WAFR*, vol. 107, 2015, pp. 461–478.
- [10] V. Lumelsky, "Sensing, intelligence, motion: How robots and humans move in an unstructured world," *Wiley Encyclopedia of Electrical and Electronics Engineering*, pp. i–xxiv, 2005.
- [11] M. P. Strub and J. D. Gammell, "Adaptively Informed Trees (AIT*) and Effort Informed Trees (EIT*): Asymmetric bidirectional sampling-based path planning," *IJRR*, vol. 41, no. 4, pp. 390–417, 2022.
- [12] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," *IROS*, 2002.
- [13] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs," *ICRA*, 2006.
- [14] J. van den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *ICRA*, 2011, pp. 3475–3482.
- [15] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2022.
- [16] A. Zelinsky, "A mobile robot exploration algorithm," *T-RA*, 1992.
- [17] A. Stentz, "The focussed D* algorithm for real-time replanning," in *IJCAI*, 1995.
- [18] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *ICRA*, 2002, pp. 968–975.
- [19] —, "Fast replanning for navigation in unknown terrain," *T-RO*, vol. 21, no. 3, pp. 354–363, 2005.
- [20] J. Lim, M. Ghanei, R. C. Lawson, S. Srinivasa, and P. Tsiotras, "Lazy incremental search for efficient replanning with bounded suboptimality guarantees," *IJRR*, 2024.
- [21] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Department of Computer Science, Iowa State University, Tech. Rep., 1998.
- [22] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *IJRR*, vol. 30, no. 7, pp. 846–894, 2011.
- [23] J. Qi, H. Yang, and H. Sun, "MOD-RRT*: A sampling-based algorithm for robot path planning in dynamic environment," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 8, pp. 7244–7251, 2021.
- [24] J. J. Kuffner Jr. and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *ICRA*, 2000, pp. 995–1001.
- [25] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics and Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012, <https://ompl.kavrakilab.org>.
- [26] J. D. Gammell, M. P. Strub, and V. N. Hartmann, "Planner Developer Tools (PDT): Reproducible experiments and statistical analysis for developing and testing motion planners," in *Workshop on Evaluating Motion Planning Performance (EMPP), IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 23 Oct. 2022.
- [27] V. N. Hartmann, M. P. Strub, M. Toussaint, and J. D. Gammell, "Effort Informed Roadmaps (EIRM*): Efficient asymptotically optimal multiquery planning by actively reusing validation effort," in *ISRR*, vol. 27, 25–30 Sep. 2022, pp. 555–571.