2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Flexible Informed Trees (FIT*): Adaptive Batch-Size Approach in Informed Sampling-Based Path Planning

Liding Zhang[1], Zhenshan Bing[1], Kejia Chen[1], Lingyun Chen[1], Kuanqi Cai[1], Yu Zhang[1], Fan Wu[1], Peter Krumbholz[2], Zhilin Yuan[2], Sami Haddadin[1‡], Alois Knoll[1†]

***Abstract*— In path planning, anytime almost-surely asymptotically optimal planners dominate the benchmark of sampling-based planners. A notable example is Batch Informed Trees (BIT*), where planners iteratively determine paths to batches of vertices within the exploration area. However, utilizing a consistent batch size is inefficient for initial pathfinding and optimal performance, it relies on effective task allocation. This paper introduces Flexible Informed Trees (FIT*), a sampling-based planner that integrates an adaptive batch-size method to enhance the initial path convergence rate. FIT* employs a flexible approach in adjusting batch sizes dynamically based on the inherent dimension of the configuration spaces and the hypervolume of the n-dimensional hyperellipsoid. By applying dense and sparse sampling strategy, FIT* improves convergence rate while finding successful solutions faster with lower initial solution cost. This method enhances the planner's ability to handle confined, narrow spaces in the initial finding phase and increases batch vertices sampling frequency in the optimization phase. FIT* outperforms existing single-query, sampling-based planners on the tested problems in $\mathbb{R}^2$ to $\mathbb{R}^8$, and was demonstrated on a real-world mobile manipulation task.**

I. Introduction

Path planning is essential in robotics, autonomous navigation, robot manipulation, and bio-inspired robots [1]. Early methods like Dijkstra's algorithm [2] found the shortest path in a graph, while A* [3] integrated graph and heuristic search for faster pathfinding. Anytime Repairing A* (ARA*) [4] offers progressive solutions during execution, catering to real-time scenarios with its *Anytime* feature. Its *Repairing* aspect also refines solutions by adjusting search parameters.

Using graph-based planning in continuous state spaces is challenging due to the need for suitable discretization, known as *prior discretization*. Coarse resolution boosts efficiency but may yield suboptimal paths. Finer resolution improves path quality [5] but demands exponentially more computation, especially in high-dimensional environments, which are known as *the curse of dimensionality* [6]. To tackle this issue, sampling-based planning methods like Rapidly-exploring Random Trees (RRT) [7] and Probabilistic Roadmaps (PRM) [8] construct feasible paths by randomly sampling vertices in the *configuration space* (*$\mathcal{C}$-space*) and connects samples using local planners with collision checks [9]. These approaches are suitable for high-dimensional environments.

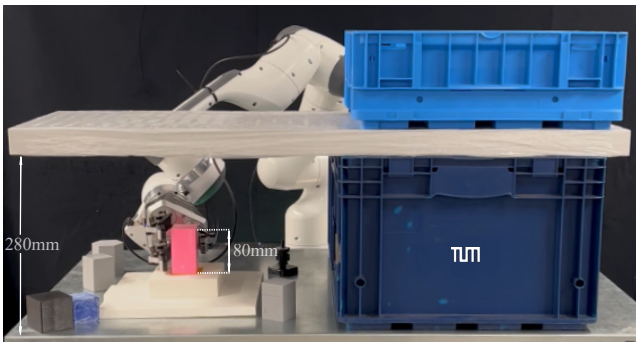


Fig. 1: FIT* on mobile manipulator robot during a real-time pull-out and place task in supermarket cell (Section VI-B).

Efficient pathfinding with high-dimensional *$\mathcal{C}$-space* in an incremental planner hinges on balancing exploration and exploitation. Batch Informed Trees (BIT*) [10], [11] compactly groups states into an implicit *random geometric graph* (RGG) [12], employing step-wise search akin to Lifelong Planning A* (LPA*) [13] based on expected solution quality. Advanced BIT* (ABIT*) [14] uses inflation and truncation factors to balance the exploration and exploitation of an increasingly dense RGG approximation. Adaptively Informed Trees (AIT*) [15], [16] utilized an asymmetrical search method with sparse collision checks in the reverse search. Effort Informed Trees (EIT*) [15], a state-of-the-art planner uses admissible cost (i.e., lower bound on the true value) and effort heuristics to enhance its capability to tackle intricate scenarios, particularly in objectives with obstacle clearance, which is not estimable with the Euclidean distance heuristic. However, these planners lack adaptability in selecting an optimal batch size during the planning process. Optimizing batch size is advisable, considering variations among robots, scenarios, dimensions, and state spaces. Fewer samples might lower the probability of sampling vertices in narrow corridors. After the *informed set* [17] (Fig. 2) contracts, more samples might lead to a more time-consuming edge check. This limitation hampers overall planning efficiency [11].

This paper presents Flexible Informed Trees (FIT*), which integrates the strengths of both *search strategies* and *approximation techniques*. FIT* integrate adaptive batch-size features to tackle narrow corridor challenges in path planning. By employing decay-based sigmoid functions to leverage the decay factor for dynamic batch size adjustments, a smoothing operation is integrated to prevent large differences in

[1]L. Zhang, Z. Bing, K. Chen, L. Chen, K. Cai, Y. Zhang, F. Wu, S. Haddadin and A. Knoll are with the Department of Informatics, Technical University of Munich, Germany. liding.zhang@tum.de

[2]P. Krumbholz, Z. Yuan are with the Department of Technology & Innovation, KION Group Linde Material Handling GmbH, Germany.

[†]The authors acknowledge the financial support by the Bavarian State Ministry for Economic Affairs, Regional Development and Energy (StMWi) for the Lighthouse Initiative KI.FABRIK (Phase 1: Infrastructure as well as the research and development program under grant no. DIK0249).

[‡]The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany (BMBF) in the programme of "Souverän. Digital. Vernetzt." Joint project 6G-life, project identification number 16KISK002.

Corresponding author: Zhenshan Bing.

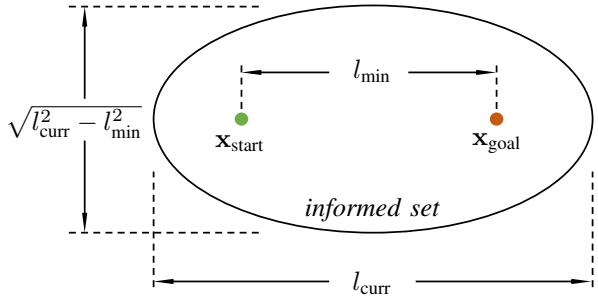


Fig. 2: The 2D representation of hyperellipsoid, its shape is determined by the start, goal states, and two key costs: the theoretical minimum cost l_{min} and the current best solution cost l_{curr}. The eccentricity of the hyperellipsoid is given by the ratio $l_{\text{min}}/l_{\text{curr}}$ [19].

resizing steps while maintaining optimality. This integration enhances the efficiency of the approximation process. The practical efficacy of FIT* has been thoroughly demonstrated through real-world applications, as depicted in Fig. 1 and 5. We evaluated autonomous mobile manipulator navigation performance. Its adaptability in dynamically adjusting batch sizes during optimization demonstrated notable enhancements in computational time efficiency and solution quality. FIT* addresses the limitations of existing informed heuristic sampling-based algorithms by computing appropriate batch size during path optimization.

The contributions of this work are summarized as follows:

- *Efficient initial pathfinding*: FIT* uses dense sampling strategy in the initial pathfinding phase tackles difficult-to-sample regions problem, reducing the initial pathfinding time in various test environments up to approx. 24%.
- *Batch sampling frequency*: The sparse sampling strategy in the path-optimized phase benefits the search process by frequently sampling points in the informed set. Leads to a higher probability of sampling *key* points.
- *Adaptability to confined environments*: FIT* was tested in real-world applications, including mobile manipulator pull-out and place tasks in narrow spaces. FIT* showed a 27.78% solution cost reduction in experiments.

II. Related Work

In sampling-based motion planning [18], two strategies for sampling vertices in the $\mathcal{C}$*-space* are *Dense sampling* and *Sparse sampling* (Fig. 3). Dense sampling generates more samples per batch, increasing the likelihood of including critical vertices in narrow corridors (key sample areas) but requiring more time for RGG construction and edge checking. Sparse sampling generates fewer samples per batch, expediting RGG construction and edge checking but reducing the likelihood of sampling critical vertices. Despite this trade-off, sparse sampling facilitates quicker transitions to subsequent sampling iterations.

Effective motion planning involves striking a balance between *exploration* and *exploitation*. Overemphasizing exploration wastes time mapping the environment without progressing toward the goal, while excessive exploitation may overlook superior solutions [20]. Exploring/Exploiting Trees (EET) [21] aims to strike this balance by integrating gradient information. RRT-Connect [22] extends RRT to efficiently find a path between two points in $\mathcal{C}$-space by utilizing two RRTs to connect start and goal states. RRT* [20] enhances optimality by incrementally constructing a tree from an initial state. Informed-RRT* [19], extensions of RRT*, use heuristics or informed sampling (Fig. 2) to guide state space exploration, biasing sampling towards regions likely to contain the optimal path. Lazy-PRM [23] improves roadmap construction by deferring collision checking until necessary.

BIT* is a sampling-based planner designed to achieve almost-surely asymptotic optimality. It utilizes a *batch processing approach*, takes numerous state batches and approximates as a progressively denser edge-implicit RGG. It optimizes the tree structure iteratively to minimize computational overhead. EIT* and AIT* consist of forward edge build and reverse search to employ distinct heuristic approaches for pathfinding. AIT* excels in precision by updating its heuristic with high accuracy to match the ongoing approximation. EIT* is built on BIT* and uses effort as an additional heuristic function to estimate the collision check of the edges. In contrast, EIT* focuses on optimizing the pathfinding process by utilizing the count (e.g., *number of collision checks*) as an effort estimate heuristic, leading to efficient path discovery. Through iterative optimization, they approximate a near-optimal path, effectively balancing exploration and exploitation for efficient robot navigation.

A. Ellipsoid in high-dimensional space

An ellipsoid in an n-dimensional Euclidean space $\mathbb{R}^n$ is defined as the locus of all points $\mathbf{x}$ that satisfy the equation:

$$\left(\frac{x_1}{v_1}\right)^2 + \left(\frac{x_2}{v_2}\right)^2 + \cdots + \left(\frac{x_n}{v_n}\right)^2 = 1, \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ represents the coordinates of a point on the ellipsoid, and $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ is a given vector in $\mathbb{R}^n$. The components of the vector $\mathbf{v}$ correspond to the lengths of the semi-axes of the ellipsoid along each coordinate axis in $\mathbb{R}^n$ [24].

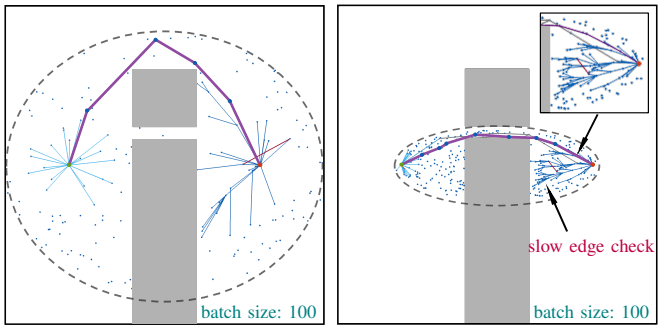
Open Motion Planning Library (OMPL) [25], is a widely used open-source database designed to address motion planning challenges in robotics and related fields. It provides a comprehensive framework and a suite of tools to assist researchers and developers in developing efficient motion planning algorithms. We integrated the proposed algorithm, Flexible Informed Trees (FIT*), into the OMPL framework and Planner-Arena benchmark database [26], along with Planner Developer Tools (PDT) [27].

III. Problem Formulation

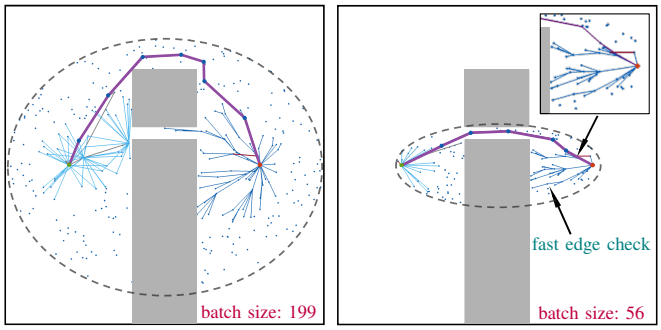
We define the problem of optimal planning in a manner associated with the definition provided in [20].

Problem Definition 1 (Optimal Planning): Consider a planning problem with the state space $X \subseteq \mathbb{R}^n$. Let $X_{\text{obs}} \subset X$ represent states in collision with obstacles, and $X_{\text{free}} = cl(X \setminus X_{\text{obs}})$ denote the resulting permissible states, where $cl(\cdot)$ represents the *closure* of a set. The initial state is denoted by $\mathbf{x}_{\text{start}} \in X_{\text{free}}$, and the set of desired final states is $X_{\text{goal}} \subset X_{\text{free}}$. A sequence of states $\sigma : [0, 1] \mapsto X$ forms a continuous map (i.e., a collision-free, feasible path), and Σ represents the set of all nontrivial paths.

The optimal solution, represented as σ^*, corresponds to the path that minimizes a selected cost function $s : \Sigma \mapsto \mathbb{R}_{\geq 0}$.



(a) EIT*-initial phase (sparse) (b) EIT*-optimal phase (dense)



(c) FIT*-initial phase (dense) (d) FIT*-optimal phase (sparse)

Fig. 3: Four snapshots of how EIT* and FIT* place their sampling strategy. FIT* employs an adaptive batch size (1 to 199) strategy. EIT* maintains a constant batch size (100) across all sample batches. The approach contains a combination of sparse and dense sampling, where more samples are utilized to expedite tree construction (c) compared to (a). Subsequently, FIT* reduces the samples per batch, resulting in less time-consuming edge checking and an increased frequency of batch updating (d) compared to (b).

This path connects the initial state $\mathbf{x}_{\text{start}}$ to any goal state $\mathbf{x}_{\text{goal}} \in X_{\text{goal}}$ through the free space:

$$\sigma^* = \arg\min_{\sigma \in \Sigma} \{ s(\sigma) | \sigma(0) = \mathbf{x}_{\text{start}}, \sigma(1) \in \mathbf{x}_{\text{goal}}, \forall t \in [0,1], \sigma(t) \in X_{\text{free}} \}, \tag{2}$$

where $\mathbb{R}_{\geq 0}$ denotes non-negative real numbers. The cost of the optimal path is s^*.

Considering a discrete set of states, $X_{\text{samples}} \subset X$, as a graph where edges are determined algorithmically by a transition function, we can describe its properties using a probabilistic model implicit dense RGGs when these states are randomly sampled, i.e., $X_{\text{samples}} = \{\mathbf{x} \sim \mathcal{U}(X)\}$, as discussed in [12].

The characteristics of the anytime almost-surely sampling-based planner with the definition are provided in [17].

Problem Definition 2 (Almost-sure asymptotic optimality): A planner is considered almost-surely asymptotically optimal as the number of samples tends to *infinity* which covers the entire general $\mathcal{C}$*-space*. In this scenario, if an optimum solution exists (as definition 1 optimal planning), the probability of the planner asymptotically converging to the optimal solution equals one when sample size q goes infinite.

$$P\left(\limsup_{q \to \infty} c(\sigma_q) = c(\sigma^*) \right) = 1. \tag{3}$$

where q represents the number of samples a planner has sampled, σ_q signifies the path derived by the planner from those batch of samples, σ^* stands for the optimal solution to the planning problem, and $c(\cdot)$ denotes the path's cost at the informed batch. After discovering an initial solution, the planner utilizes the remaining computational time to optimize the path quality of the existing solution.

Algorithm 1: Asymmetric bidirectional search

```
repeat
    improve RGG approximation (re-sampling)
    update current batch size (adaptive adjust)
    calculate heuristics (reverse search)
    while RGG approximation is beneficial do
        discover the feasible path (forward search)
        if found invalid edge or unprocessed state
            update heuristics (reverse search)
until planner termination condition
```

IV. Flexible Informed Trees (FIT*)

FIT* builds on EIT* and follows the asymmetric bidirectional search process with adaptive batch-size tunning (Alg. 1). Moreover, FIT* dynamically adjusts batch sizes based on the state space's geometry, fine-tuning the sampling density according to the dimension of $\mathcal{C}$*-space* and hypervolume of the n-dimensional hyperellipsoid. This difference in sampling strategy leads to distinct efficiencies and computational performance within planning algorithms.

A. Notation

The state space of the planning problem is denoted by $X \subseteq \mathbb{R}^n$, where $n \in \mathbb{N}$. The start point is represented by $\mathbf{x}_{\text{start}} \in X$, and the goals are denoted by $X_{\text{goal}} \subset X$. The sampled states are denoted by X_{sampled}. The forward and reverse search trees are represented by $\mathcal{F} = (V_{\mathcal{F}}, E_{\mathcal{F}})$ and $\mathcal{R} = (V_{\mathcal{R}}, E_{\mathcal{R}})$, respectively. The vertices in these trees, denoted by $V_{\mathcal{F}}$ and $V_{\mathcal{R}}$, are associated with valid states. The edges in the forward tree, $E_{\mathcal{F}} \subset V_{\mathcal{F}} \times V_{\mathcal{F}}$, represent valid connections between states, while the edges in the reverse tree, $E_{\mathcal{R}} \subset V_{\mathcal{R}} \times V_{\mathcal{R}}$, may traverse invalid regions of the problem domain. An edge comprises a source state, $\mathbf{x}_s$, and a target state, $\mathbf{x}_t$, denoted as $(\mathbf{x}_s, \mathbf{x}_t)$. $\mathcal{Q}_{\mathcal{F}}$ and $\mathcal{Q}_{\mathcal{R}}$ designate the edge-queue for the forward search and reverse search, respectively. The true connection cost between two states in $\mathcal{C}$*-space* is represented by the function $c : X \times X \to [0, \infty)$.

For sets A, B, and C with B, C being subsets of A, the notation $B \xleftarrow{+} C$ denotes $B \leftarrow B \cup C$, and $B \xleftarrow{-} C$ denotes $B \leftarrow B \setminus C$.

FIT-specific Notation:* Incorporating attenuation introduces a decay factor Ψ_{decay}, with specified minimal $m_{\text{min}} := 1$ and maximal m_{max} sample numbers per batch. The initial batch sizes, denoted as $\mathcal{M}_{\text{initial}}$, and the current count of states sampled per batch represented as $\mathcal{M}(\Psi_{\text{current}})$. The Lebesgue measure within an n-dimensional unit ball is denoted by ζ_n. Non-negative scalar ξ_n represents the raw ratio of the initial hypervolume (i.e., Lebesgue measure) of the n-dimensional hyperellipsoid $\mathcal{V}_{\text{initial}}$ to the current hypervolume $\mathcal{V}_{\text{current}}$. A nature logarithmic tuning parameter Λ_{tuning} regulates the rate at which the ratio decreases. $\mathcal{O}_{\text{smooth}}$ represents a smoothed value after the attenuation of the initial and optimal state.

Algorithm 2: Flexible Informed Trees (FIT*)

$c_{current} \leftarrow \infty$; $\Psi_{decay} \leftarrow \infty$; $\Lambda_{tuning} \leftarrow$ *initialized*
$\mathcal{M}_{initial} \leftarrow \mathcal{M}(\Psi_{current})$; $\mathcal{V}_{current} \leftarrow \mathcal{V}_{initial}$; $\xi_{initial} := 1$ ▷ initial batch size
$X_{sampled} \leftarrow X_{goal} \cup \{\mathbf{x}_{start}\}$
$V_{\mathcal{F}}, E_{\mathcal{F}}, \mathcal{Q}_{\mathcal{F}} \leftarrow$ `expand`$(\mathbf{x}_{start})$
$V_{\mathcal{R}}, V_{\mathcal{R},closed}, E_{\mathcal{R}}, \mathcal{Q}_{\mathcal{R}} \leftarrow$ `expand`(X_{goal})
Initialize and update the Inflation Factor and Sparse Resolution
repeat
  $\xi_n \leftarrow$ `updateRawRatio`$(\mathcal{V}_{current}, \mathcal{V}_{initial})$
  $\mathcal{O}_{smooth} \leftarrow$ `sigmoidFunction`(ξ_n) ▷ update smoothed raw ratio
  $\Psi_{decay} \leftarrow$ `updateDecayFactor`$(\mathcal{O}_{smooth}, \Lambda_{tuning})$
  if *Forward queue has better edge priority than Reverse queue*
  or *target of optimal edge in forward queue*
    $(\mathbf{x}_s, \mathbf{x}_t) \leftarrow argmin_{(\mathbf{x}_s,\mathbf{x}_t)\in\mathcal{Q}_{\mathcal{R}}} \{key_{\mathcal{R}}^{FIT*}(\mathbf{x}_s, \mathbf{x}_t)\}$
      ▷ heuristics of solution cost and effort with batch-size tuning
    $\mathcal{Q}_{\mathcal{R}} \xleftarrow{-} (\mathbf{x}_s, \mathbf{x}_t)$
    $V_{\mathcal{R},closed} \xleftarrow{+} \mathbf{x}_s$
    if *No sparse collisions detected between* $(\mathbf{x}_s, \mathbf{x}_t)$
      Update cost and reverse tree structures
      $\mathcal{M}(\Psi_{current}) \leftarrow$ `adaptiveBactchSize`$(\Psi_{decay}, m_{min}, m_{max})$
    else
      Execute sparse collision case processing
  else if *Better edge in forward queue compared to current solution*
    $(\mathbf{x}_s, \mathbf{x}_t) \leftarrow$ `getBestForwardEdge`$(\mathcal{Q}_{\mathcal{F}})$
    $\mathcal{Q}_{\mathcal{F}} \xleftarrow{-} (\mathbf{x}_s, \mathbf{x}_t)$
    if *Edge is in existing forward tree*
      $\xi_n \leftarrow$ `updateRawRatio`$(\mathcal{V}_{current}, \mathcal{V}_{initial})$
      $\mathcal{O}_{smooth} \leftarrow$ `sigmoidFunction`(ξ_n)
      Update forward tree structures
    else if *Edge improves existing path*
      if *No dense collisions between* $(\mathbf{x}_s, \mathbf{x}_t)$
        Update cost and forward tree structures
        $\Psi_{decay} \leftarrow$ `updateDecayFactor`$(\mathcal{O}_{smooth}, \Lambda_{tuning})$
      else
        Execute dense collision case processing
  else
    Prune sampled states
    $\mathcal{M}(\Psi_{current}) \leftarrow$ `adaptiveBactchSize`$(\Psi_{decay}, m_{min}, m_{max})$
until *planner termination condition*

Algorithm 3: adaptiveBactchSize $(\Psi_{\text{decay}}, m_{\text{min}}, m_{\text{max}})$

$m_{min} := 1$, $m_{max} := 2m_{current} - m_{min}$
if $c_{current} \neq c_{last}$ **or** $c_{last} := \infty$
  if $c_{current} := \infty$
    return *null* ▷ ensure safety when initial solution not found yet
  $c_{last} \leftarrow$ `updateSolutionCost` $(c_{current})$
  if *pragma once*
    $\mathcal{V}_{initial} \xleftarrow{+}$ `ellipseVolumCal`(c_{last}) ▷ $\mathcal{V}_{\text{initial}}$ only calculated once
  $\mathcal{V}_{current} \xleftarrow{+}$ `ellipseVolumCal`$(c_{current})$ ▷ $\mathcal{V}_{\text{current}}$ update every-time
  $\xi_n \leftarrow$ `updateRawRatio`$(\mathcal{V}_{current}, \mathcal{V}_{initial})$
  $\mathcal{O}_{smooth} \leftarrow$ `sigmoidFunction`(ξ_n) ▷ equation (7)
  $\Psi_{decay} \leftarrow$ `updateDecayFactor`$(\mathcal{O}_{smooth}, \Lambda_{tuning})$ ▷ equation (6)
  $\mathcal{M}(\Psi_{current}) := m_{min} + \Psi_{decay}(m_{max} - m_{min})$ ▷ adapted batch size
  return $\mathcal{M}(\Psi_{current})$

TABLE I: Decay-based tuning methods comparison (100 runs)

	initial time			initial cost			success
	t_{init}^{min}	t_{init}^{med}	t_{init}^{max}	c_{init}^{min}	c_{init}^{med}	c_{init}^{max}	
*EIT**	0.0027	0.0048	∞	0.6526	1.0442	∞	0.97
FIT-B*	0.0027	0.0035	0.0083	0.6333	0.6747	1.2486	1.00
FIT-I*	0.0031	0.0049	∞	0.6471	0.7459	∞	0.97
FIT-L*	0.0028	0.0037	∞	0.6441	0.6956	∞	0.99
FIT-P*	0.0028	0.0034	0.0077	0.6406	0.6753	1.2502	1.00
FIT*-SL	0.0026	0.0030	0.0062	0.6333	0.6603	1.0608	1.00

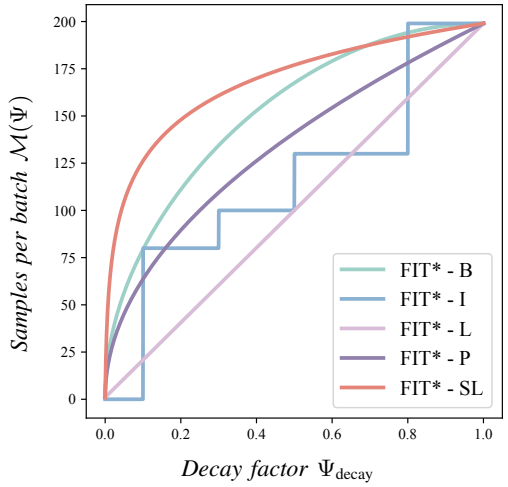


Fig. 4: This graph shows the decay-based method comparison, where the maximal batch size is 199, and the minimal batch size is 1; more specific illustrations are given in section IV-C.

B. Approximation

FIT* employs informed sampling strategies [17] to concentrate its RGG approximation on the pertinent region of the state space and dynamically adjusts the connection radius as more states are sampled. The radius (r) is updated according to the approach proposed in [20], utilizing the measure of the informed set as introduced in [17].

$$r(q) = \eta \left(2\left(1+\frac{1}{n}\right)\left(\frac{\lambda(X_{\hat{f}})}{\zeta_n}\right)\left(\frac{\log(q)}{q}\right)\right)^{\frac{1}{n}}, \quad (4)$$

here, q denotes the number of sampled states in the informed set, $\eta > 1$ is a tuning parameter, $\lambda(\cdot)$ denotes the Lebesgue measure, and n represents the dimension of the state space.

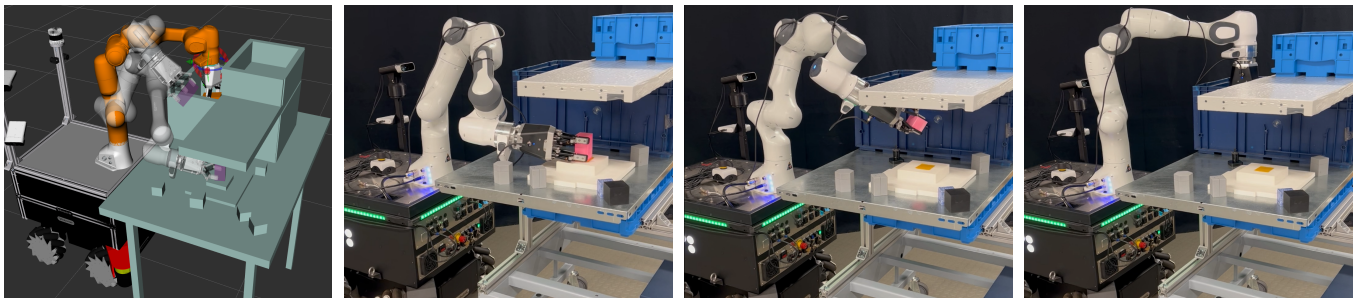
C. Adaptive Batch-Size

FIT* dynamically adjusts the number of samples in batch size. Specifically, it leverages the sigmoid function in conjunction with logarithmic (*sigmoid-log*, FIT*-SL). As observed in Table I, where t_{init}^{min} represents the minimal initial planning time over 100 runs, and c_{init}^{max} represents the maximal initial cost of the planning problem, respectively. The *cost* and *time* of unsuccessful attempts are represented as infinity.

Within the FIT*'s flexible batch size context (Alg. 2 and 3), our comparison focused on exploring various decay methods to comprehend their distinct impacts (Table I and Fig. 4). These decay methods encompassed *linear* decay (FIT*-L), *brachistochrone* curve-based decay (FIT*-B), decay functions following an *parabola* pattern (FIT*-P), and decay based on the *iteration count* (FIT*-I). Each method was thoroughly examined 100 runs to assess its effectiveness in shaping the decay factor Ψ_{decay}. Through experimentation, it is concluded that FIT*-SL stands out as the most efficient approach, demonstrating the quickest *median time* and shortest *median cost* for the initial solution. It strikes a balance between adaptability to denser sampling in the initial pathfinding phase and sparse sampling during the optimization phase of $\mathcal{C}$-space.

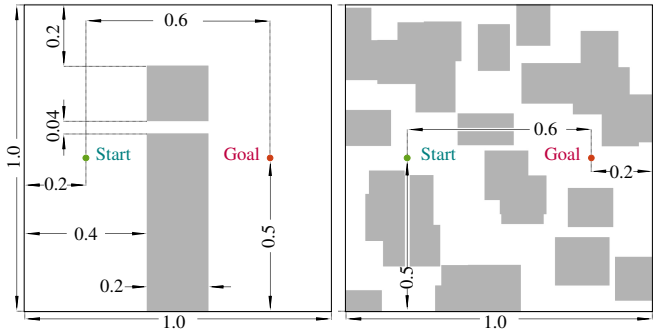
V. Formal Analysis

In this paper, we refer to Definition 24 from [20] to establish the concept of almost-sure asymptotic optimality.



(a) Simulation scenery (b) Start configuration (c) Transition configuration (d) Goal configuration

Fig. 5: Illustrates the simulation (a) and the real-world scenarios of DARKO robot for the intralogistics task, (b) shows the start configuration of the arm in position to pick up the red cube from the metal sheet table, (c) shows the transition configuration position of the task, (d) shows the goal configuration of the arm in position to place a cube in the box.



(a) Wall Gap (b) Random Rectangles

Fig. 6: The 2D representation of the simulated planning problems in Section VI. The state space, denoted as $X \subset \mathbb{R}^n$, is constrained within a hypercube with one width for both problem instances. Specifically, we conducted ten distinct instantiations of the random rectangles experiment and the outcomes are showcased in Fig. 7.

A. Almost-Sure Asymptotically Optimal Path

The FIT* algorithm utilizes an RGG approximation similar to EIT*, with the underlying graph likely to contain an asymptotically optimal path. The graph search in FIT* shows asymptotic resolution optimality. Given EIT*'s status as an almost-surely asymptotically optimal algorithm [27], it indicates that this RGG approximation includes an asymptotically optimal path. Therefore, the adaptive batch size computation affects the number of edge collision checks and does not effect almost-sure asymptotic optimality.

B. Statistical Formulation and Hypotheses

The sampling-based planner FIT* dynamically adjusts batch sizes, increasing samples in the initial solution stage for faster convergence to a feasible path. Conversely, reducing samples per batch expedites optimization by minimizing edge checks, accelerating the batch sample update frequency, and reducing computational time. The *adaptiveBatchSize* function $\mathcal{M}(\Psi)$ is the major influence, considering the integration of the decay factor. It reflects how the expansion and contraction of the hyperellipsoids in different dimensions affect the appropriate batch size, defined as follows:

$$\mathcal{M}(\Psi) := m_{\text{min}} + (m_{\text{max}} - m_{\text{min}}) \times \Psi_{\text{decay}}, \tag{5}$$

Natural logarithm smoothly diminishes the decay rate, averting sudden drops for stable model optimization. The decay factor Ψ_{decay} is determined through the division operation involving the logarithmic function. This relationship emphasizes how the dimensions of the hyperellipsoid influence the exploration of the state space in a logarithmic manner.

$$\Psi_{\text{decay}} = \frac{\ln(1 + \Lambda \times \mathcal{O}_{\text{smooth}})}{\ln(1 + \Lambda)}, \tag{6}$$

The sigmoid smoothing technology of the raw ratio $\mathcal{O}_{\text{smooth}}$ ensures a gradual transition between batch sizes, mirroring how the hyperellipsoids's hypervolume changes smoothly with varying semi-axes lengths. This smoothing function allows for a more continuous adjustment of the batch size.

$$\mathcal{O}_{\text{smooth}} = \frac{1}{1 + e^{-10 \times (\xi_n - 0.5)}}, \tag{7}$$

where e is Euler's number $e = \sum_{i=0}^{\infty} \frac{1}{n!}$

Decay tuning parameter Λ_{tuning} is correlated on the problem's dimensionality $n_{\text{dimension}}$ along with the m_{min} and m_{max} samples. This tuning parameter is a composite representation that makes the batch size related to *C-space* dimensional information and becomes more problem-specific during adaptation.

$$\Lambda_{\text{tuning}} = \frac{\mathcal{M}(\Psi_{\text{max}}) + \mathcal{M}(\Psi_{\text{min}})}{n_{\text{dimension}}}, \tag{8}$$

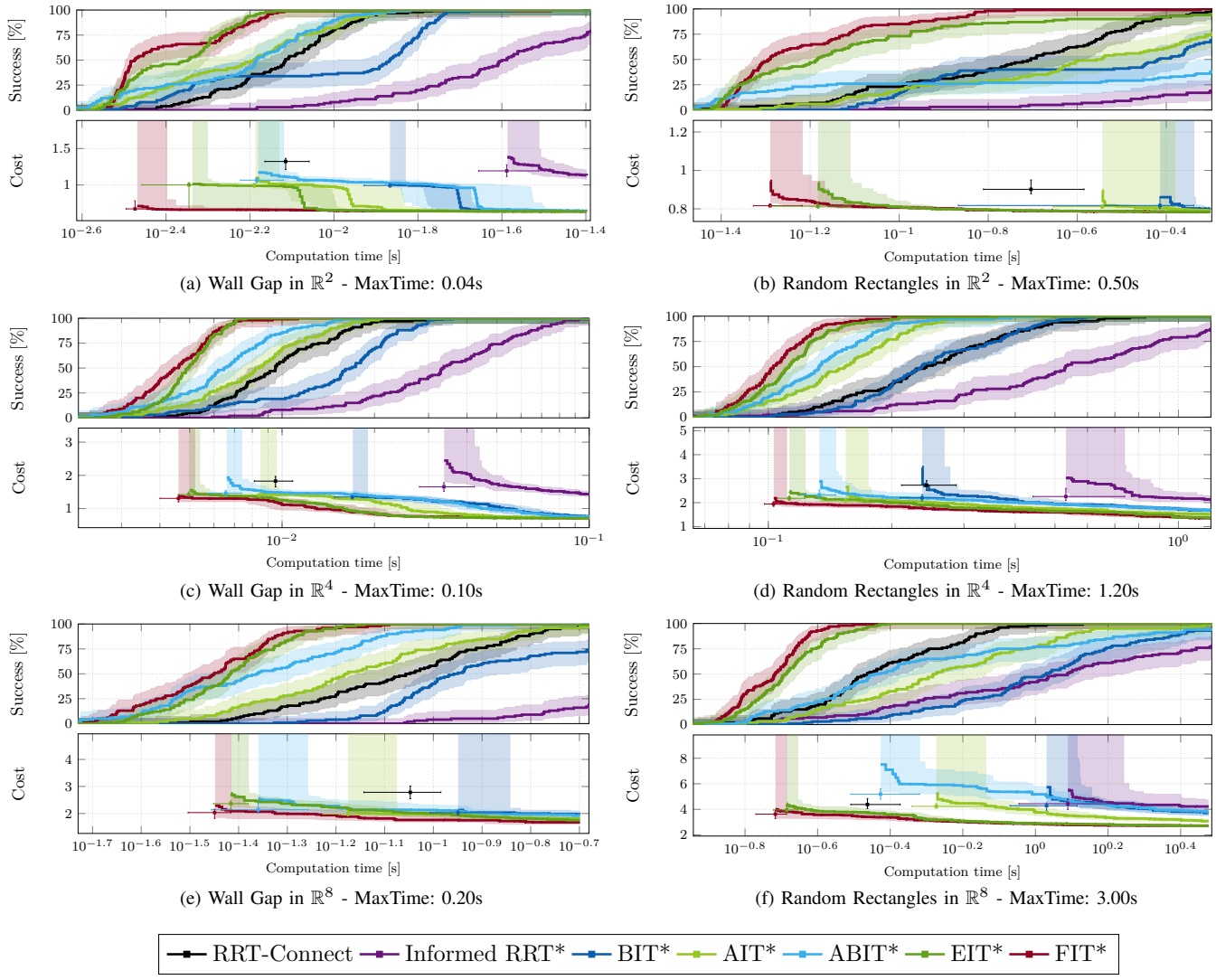
The adjustment of batch size responds to the current raw ratio ξ_n, aligning with the optimization phase of the problem. This ratio is affected by the contraction (i.e., solution cost update) of the hypervolume of the current n-dimensional hyperellipsoids, this raw ratio is represented as:

$$\xi_n = \frac{\mathcal{V}_{\text{current}}}{\mathcal{V}_{\text{initial}}}. \tag{9}$$

Given the continuous improvement of the solution cost, the hypervolume shrinks accordingly, $\mathcal{V}_{\text{current}}$ consistently remains less than or equal to $\mathcal{V}_{\text{initial}}$. Consequently, the current raw ratio of the process ξ_n resides within the interval $(0, 1]$.

VI. Experimental Results

FIT* was tested against existing algorithms in both simulated random scenarios (Fig. 6) and real-world manipulation problems (Fig. 5). The comparison involved several versions of RRT-Connect, Informed RRT*, BIT*, AIT*, ABIT*, and EIT* sourced from the Open Motion Planning Library (OMPL) [25]. The evaluations are implemented on



(a) Wall Gap in $\mathbb{R}^2$ - MaxTime: 0.04s

(b) Random Rectangles in $\mathbb{R}^2$ - MaxTime: 0.50s

(c) Wall Gap in $\mathbb{R}^4$ - MaxTime: 0.10s

(d) Random Rectangles in $\mathbb{R}^4$ - MaxTime: 1.20s

(e) Wall Gap in $\mathbb{R}^8$ - MaxTime: 0.20s

(f) Random Rectangles in $\mathbb{R}^8$ - MaxTime: 3.00s

Fig. 7: Detailed experimental results from Section VI-A are presented above. Fig. (a), (c) and (e) depict test benchmark wall gap outcomes in $\mathbb{R}^2$, $\mathbb{R}^4$ and $\mathbb{R}^8$, respectively. Panel (b) showcases ten random rectangle experiments in $\mathbb{R}^2$, while panels (d) and (f) demonstrate in $\mathbb{R}^4$ and $\mathbb{R}^8$. In the cost plots, boxes represent solution cost and time, with lines showing cost progression for an almost surely optimal planner (unsuccessful runs have infinite cost). Error bars provide nonparametric 99% confidence intervals for solution cost and time.

a computer with an Intel i7 3.90 GHz processor and 32GB of LPDDR3 3200 MHz memory. These comparisons were carried out in simulated environments ranging from $\mathbb{R}^2$ to $\mathbb{R}^8$. The primary objective for the planners was to minimize path length. The RGG constant η was uniformly set to 1.001, and the rewire factor was set to 1.2 for all planners.

In the case of RRT-based algorithms, a goal bias of 5% was employed, and the maximum edge lengths were appropriately determined based on the dimensionality of the space. Meanwhile, BIT*, AIT*, ABIT*, and EIT* maintained a fixed sampling of 100 states per batch, regardless of the dimensionality of the state space. These planners also had graph pruning deactivated and utilized the Euclidean distance and effort as heuristic functions. FIT*'s adaptive batch size technology dynamically adjusts the batch size from 1 to 199 (i.e., average of 100) batch sizes (Fig. 4). The number of samples at anytime is determined by the planner's adaptive mechanisms, ensuring quantities that are optimized for each solution cost update and batch re-sampling process.

A. *Experimental Tasks*

The planners were subjected to testing across three distinct problem domains: $\mathbb{R}^2$, $\mathbb{R}^4$, and $\mathbb{R}^8$. In the first scenario, a constrained environment resembling a wall with a narrow gap was simulated, allowing valid paths in two general directions for non-intersecting solutions (Fig. 6a). The path planning objective is to optimize the initial and final path length rapidly. Each planner underwent 100 runs, and the computation time for each asymptotically optimal planner is demonstrated in the labels, with varying random seeds. The overall success rates and median path lengths for all planners are depicted in Fig. 7a, 7c, and 7e. In the second test scenario, random widths were assigned to *axis-aligned hyperrectangles*, generated arbitrarily within the $\mathcal{C}$*-space* (Fig. 6b). Random rectangle problems were created for each dimension of the $\mathcal{C}$*-space*, and each planner underwent 100 runs for every instance. Fig. 7b, 7d, and 7f illustrate the overall success rates and median path costs within the computation time for all the planners.

FIT* employs an adaptive batch size feature, enhancing convergence time in the initial pathfinding phase by up to 24% and achieving faster solutions with lower initial costs.

B. Path Planning for DARKO

FIT* demonstrated its effective adaptive batch size techniques during a field test as part of the inventory management (Fig. 1 and Fig. 5). DARKO is a mobile manipulation robotic platform (8-DoF) created by combining Robotnik base robot and Franka Emika Panda manipulator, addressing intralogistics challenges. The intricacy of the narrow space presents challenging planning problems, primarily due to the computationally expensive state evaluations required. All planners had 1.0 seconds to address this confined, limited space pull-out and place problem. Over 15 trials, FIT* was 100% successful with a median solution cost of 19.1021. EIT* was also 100% successful but had a median solution cost of 22.6917. AIT* was 86.7% successful with a median solution cost of 26.4482, and ABIT* was 80% successful with a median solution cost of 25.6341. In contrast to other planners that rely on occupied space, FIT* showcased efficiency by saving working space and completing the task with the shortest optimal path length. The detailed behavior of DARKO can be viewed in https://youtu.be/N9HdU56v1nU.

VII. Conclusion

This paper introduces FIT*, an adaptive batch size method planner correlated to the $\mathcal{C}$-*space*'s dimensionality and the hypervolume of the n-dimensional hyperellipsoid. During the initial solution phase, increased samples per batch accelerate discovery by enhancing the probability of sampling key areas. In the optimal phase, fewer samples per batch reduce collision checking time, improving efficiency. FIT*'s adaptability was demonstrated in a real-world scenario with the DARKO robot, showcasing FIT*'s practical applicability. Future work can further test FIT* in bio-inspired robots [1] and integrate learning method [28], [29] for motion planning.

In conclusion, FIT* optimizes batch sizes dynamically using a sigmoid-log function and a decay factor, positioning it as a promising research direction in robot path planning.

References

[1] Z. Bing, A. Rohregger, F. Walter, Y. Huang, P. Lucas, F. O. Morin, K. Huang, and A. Knoll, "Lateral flexion of a compliant spine improves motor performance in a bioinspired mouse robot," *Science Robotics*, vol. 8, no. 85, 2023.

[2] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[4] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," *Advances in neural information processing systems*, vol. 16, 2003.

[5] D. Bertsekas, "Convergence of discretization procedures in dynamic programming," *IEEE Transactions on Automatic Control*, vol. 20, no. 3, pp. 415–419, 1975.

[6] R. Bellman, "Dynamic programming, princeton univ," *Press Princeton, New Jersey*, 1957.

[7] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[8] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[9] K. Cai, W. Chen, C. Wang, S. Song, and M. Q.-H. Meng, "Human-aware path planning with improved virtual doppler method in highly dynamic environments," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 1304–1321, 2022.

[10] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 3067–3074.

[11] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch informed trees (bit*): Informed asymptotically optimal anytime search," *The International Journal of Robotics Research*, vol. 39, no. 5, 2020.

[12] M. Penrose, *Random geometric graphs*. OUP Oxford, 2003, vol. 5.

[13] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning a*," *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.

[14] M. P. Strub and J. D. Gammell, "Advanced bit* (abit*): Sampling-based planning with advanced graph-search techniques," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 130–136.

[15] M. P. Strub and J. D. Gammell, "Adaptively informed trees (ait*) and effort informed trees (eit*): Asymmetric bidirectional sampling-based path planning," *The International Journal of Robotics Research*, vol. 41, no. 4, pp. 390–417, 2022.

[16] 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.

[17] 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.

[18] K. Cai, W. Chen, D. Dugas, R. Siegwart, and J. J. Chung, "Sampling-based path planning in highly dynamic and crowded pedestrian flow," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[19] 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.

[20] 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.

[21] M. Rickert, O. Brock, and A. Knoll, "Balancing exploration and exploitation in motion planning," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 2812–2817.

[22] 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.

[23] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 521–528.

[24] F. Han and H. Liu, "Eca: High-dimensional elliptical component analysis in non-gaussian distributions," *Journal of the American Statistical Association*, vol. 113, no. 521, pp. 252–268, 2018.

[25] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[26] M. Moll, I. A. Sucan, and L. E. Kavraki, "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 96–102, 2015.

[27] 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 *Proceedings of the Workshop on Evaluating Motion Planning Performance (EMPP), IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[28] Z. Bing, D. Lerch, K. Huang, and A. Knoll, "Meta-reinforcement learning in non-stationary and dynamic environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3476–3491, 2023.

[29] Z. Bing, Y. Yun, K. Huang, and A. Knoll, "Context-based meta-reinforcement learning with bayesian nonparametric models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–18, 2024.