Path Planning for Manipulation using Experience-driven Random Trees

Èric Pairet[1,2], Constantinos Chamzas[2], Yvan Petillot[1], Lydia E. Kavraki[2]

***Abstract*—Robotic systems may frequently come across similar manipulation planning problems that result in similar motion plans. Instead of planning each problem from scratch, it is preferable to leverage previously computed motion plans, i.e., experiences, to ease the planning. Different approaches have been proposed to exploit prior information on novel task instances. These methods, however, rely on a vast repertoire of experiences and fail when none relates closely to the current problem. Thus, an open challenge is the ability to generalise prior experiences to task instances that do not necessarily resemble the prior. This work tackles the above challenge with the proposition that experiences are "decomposable" and "malleable", i.e., parts of an experience are suitable to relevantly explore the connectivity of the robot-task space even in non-experienced regions. Two new planners result from this insight: experience-driven random trees (ERT) and its bi-directional version ERTConnect. These planners adopt a tree sampling-based strategy that incrementally extracts and modulates parts of a single path experience to compose a valid motion plan. We demonstrate our method on task instances that significantly differ from the prior experiences, and compare with related state-of-the-art experience-based planners. While their repairing strategies fail to generalise priors of tens of experiences, our planner, with a single experience, significantly outperforms them in both success rate and planning time. Our planners are implemented and freely available in the Open Motion Planning Library.**

***Index Terms*—Manipulation Planning; Motion and Path Planning; Learning from Experience; Autonomous Agents**

I. INTRODUCTION

A long-envisioned requisite for fully-autonomous robotic manipulation is to endow robots with the ability to learn from and improve through experiences. For example, consider a robot on a shelf stacking task (see Figure 1). Such a robot may frequently come across similar task instances that result in similar motion plans. Despite the resemblance among problems, the most common approach is to plan from scratch; neither prior information nor recurrent computations are leveraged to aid in solving related queries. This strategy can lead to unnecessary long planning times. Instead, the commonalities between instantiations should be considered as prior knowledge at the planning stage. However, this is not a trivial problem. The planner must reason over the relevant features that allow for the generalisation of prior knowledge even to notably different task instances.

Manuscript received: October 8, 2020; Revised January 14, 2019; Accepted February 14, 2021. This paper was recommended for publication by Editor Hong Liu upon evaluation of the Associate Editor and Reviewers' comments.

This research has been partially supported by the Scottish Informatics and Computer Science Alliance (SICSA), ORCA Hub EPSRC (EP/R026173/1) and consortium partners. Work by LEK and CC is supported in part by NSF 1718478 and NSF 2008720, Rice University Funds, and NSF 1842494 (CC).

[1]Edinburgh Centre for Robotics, University of Edinburgh and Heriot-Watt University (UK). eric.pairet@ed.ac.uk, y.r.petillot@hw.ac.uk

[2]Kavraki Lab, Department of Computer Science at Rice University, Houston TX (USA). chamzas@rice.edu, kavraki@rice.edu

Digital Object Identifier (DOI): see top of this page.

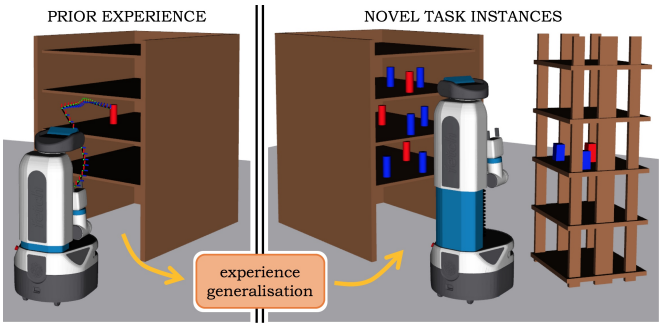


Fig. 1: Our planner can leverage a single path (experience) computed in a particular task instance, e.g., "fetch the red object" (left), to efficiently solve novel task instances (right) that remarkably differ from the experience, e.g., in obstacles (blue objects), shelf structural geometry and target locations.

Related work. Leveraging prior experiences for building motion plans efficiently has drawn special attention to the learning and planning communities. Learning-based approaches infer the underlying task policy from a given set of demonstrations, which is then used to retrieve task-related motion plans (e.g., [1]–[4]). Relevant features are extracted from the demonstrations such that the learnt policy can generalise to novel task instances. Although these methods are capable of computing plans quickly by learning from experience, they typically generalise poorly to task instances that significantly differ from those observed a priori [5].

On coping with varying task instances while leveraging experiences, sampling-based planning offers a promising venue to generalise the a priori knowledge. Such a strategy is known as experience-based planning. There are mainly two orthogonal approaches: (1) biasing the sampling into task-relevant areas, and (2) exploiting previously computed motions. This work is, in spirit, closer to the latter: leveraging prior motions. Related work is discussed for both alternatives.

(1) Biasing the sampling involves guiding the exploration towards task-relevant regions of the configuration space. A common approach is to take advantage of geometric features of the workspace to guide the sampling in the configuration space (e.g., [6]–[10]). Strategies that bias the sampling can significantly speed up queries, but they rely on identifying familiar workspace features to infer relevant samples in the configuration space. Therefore, their applicability is mainly limited to task instances that resemble those observed a priori, leading to a lack of generalisation to new environments.

(2) Using previously computed motions consists of storing experienced motions in a library (e.g., [11], [12]) or jointly as a graph (e.g., [13], [14]). These methods recall exact prior expe-

A visual aid about the experience-driven random trees planners can be found in: https://youtu.be/kD3A3Xs_psI.

riences to solve the current planning query. In Lightning [11], the most relevant experience is retrieved based on the start-goal proximity of a experience and the current query. The nearest path is chosen to be repaired. The repair step employs the bi-directional rapidly-exploring random tree (RRTConnect) to reconnect the end-points of segments originated by variant constraints, e.g., obstacles. Differently, Experience Graphs [13] build a roadmap of experiences to then search it using some heuristics. In a similar vein, Thunder [14] creates a sparse roadmap from all experiences, which is repeatedly queried via A* until a valid path is found. If the graph does not contain a valid path, candidate paths, if any, are considered for repairing. The repairing invokes RRTConnect to reconnect the disconnected states along the candidate paths. All these path-centric approaches exploit prior motion plans in the exact configuration they were experienced, i.e., "rigidly". This leads to poor performance when planning in non-experienced regions of the robot-task space. Therefore, for these methods to work, the library must contain a prior path that already resembles a valid motion plan for the current planning problem. Consequently, current experienced-based planners are dependant on a vast and extremely relevant set of prior experiences to counteract their lack of generalisation capabilities.

Contribution. In this work, we change the paradigm in which prior path experiences stored in libraries of motions are being used. Instead of exploiting prior motions "rigidly" to preserve the invariant constraints, we use them in a "decomposable" and "malleable" way to infer the next move given a particular state of the robot in a task. With this proposition, we present experience-driven random trees (ERT) and its bi-directional version ERTConnect, two experience-based planners capable of generalising a single prior motion plan across significantly varied task instances. These planners leverage a path experience by parts to iteratively build a tree of micro-experiences, i.e., segments that resemble those in the prior experience. Suitable micro-experiences result from semi-randomly morphing different parts of the experience. Such a strategy proves to be useful to efficiently explore the connectivity of the robot-task space even in non-experienced regions. Additionally, we discuss how to select the best candidate for our planner given a library of path experiences. Such experience selection strategy enables the use of our planners in frameworks that incrementally build libraries of experiences by adding newly computed motion plans (e.g., [11], [14]), as well as in systems that gather experiences from human demonstrations (e.g., [15]–[17]).

The key insight of our approach is that prior experiences are of a better use when leveraged in a "malleable" fashion, oppositely to the common "rigid" usage of experiences. Thus, contrary to prior work, the applicability of our planner goes beyond task instances that closely resemble those observed a priori. Empirical analysis demonstrates our planner's ability to leverage prior experiences efficiently and to generalise them to distinguishably dissimilar task instances. In these challenging conditions, while related state-of-the-art experience-based planners fail to exploit vast repertoires of prior path experiences, our planner, with a single path experience, significantly outperforms them in both success rate and planning time.

II. PROBLEM DEFINITION AND APPROACH OVERVIEW

In this manuscript, we are interested in families of motion planning problems that involve similar task instances and thus, seek similar motion plans. The commonalities between instantiations are of interest because they open the possibility for a robot to leverage prior information about the task. Enabling the robot to exploit such similarities would allow it to efficiently solve tasks related to those seen a priori.

Consider a robot with configuration space $\mathcal{Q} \in \mathbb{R}^n$ conducting a particular task, e.g., shelf stacking (see Figure 1). Let $\mathcal{Q}_{\text{obst}} \subset \mathcal{Q}$ be the region of the configuration space occupied by obstacles, and $\mathcal{Q}_{\text{free}} = \mathcal{Q} \setminus \mathcal{Q}_{\text{obst}}$ be the collision-free region. Let $\mathbf{q} \in \mathcal{Q}$ denote a particular robot configuration, and $\alpha \in [0, 1]$ be a phase variable that indicates the progress on the execution of a collision-free motion plan. Then, the state of the robot in a motion plan is defined in the configuration-phase space $\mathcal{S} = \mathcal{Q} \times \mathbb{R}_{[0,1]}$ as $\mathbf{s} = \langle \mathbf{q}, \alpha \rangle \in \mathbb{R}^{n+1}$. The valid regions in the configuration-phase are defined as:

$$\mathcal{S}_{\text{free}} = \{ \langle \mathbf{q}, \alpha \rangle \in \mathcal{S} \mid \mathbf{q} \in \mathcal{Q}_{\text{free}} \}. \tag{1}$$

Let $\mathcal{A}$ be some prior information about the task. In this work, we consider prior knowledge defined by a library of path experiences $\mathcal{A} = \{\xi_{\mathcal{D}_1}, \xi_{\mathcal{D}_2}, \ldots, \xi_{\mathcal{D}_j}\}$, where each $\xi_{\mathcal{D}}$ is a path (prior experience) solving a particular task instance. Paths as priors are of particular interest since they can be acquired over time from the robot's planning solutions on similar task instances, or from external sources, such as from a human kinaesthetically guiding a robot through a task. Note that the focus of this manuscript is experience-based planning, where a set of prior path experiences $\mathcal{A}$ relevant to the current problem is assumed to be provided. Therefore, given a library $\mathcal{A}$, and the start $\langle \mathbf{q}_{\text{start}}, 0 \rangle \in \mathcal{S}_{\text{free}}$ and goal $\langle \mathbf{q}_{\text{goal}}, 1 \rangle \in \mathcal{S}_{\text{free}}$ states, the motion planning problem considered in this work seeks a planning process $\mathcal{J} : \mathcal{A} \to \xi$ capable to leverage $\mathcal{A}$ to efficiently find a collision-free continuous path $\xi : \alpha \in [0, 1] \to \mathcal{S}_{\text{free}}$ that connects $\xi(0) = \mathbf{q}_{\text{start}} \in \mathcal{S}_{\text{free}}$ to $\xi(1) = \mathbf{q}_{\text{goal}} \in \mathcal{S}_{\text{free}}$.

Our approach to take advantage of a library of experiences $\mathcal{J} : \mathcal{A} \to \xi$ is twofold. First, as discussed in Section IV-A, we select a path experience $\xi_{\mathcal{D}} \in \mathcal{A}$ suitable for the current planning problem. Then, we exploit the selected prior $\mathcal{L} : \xi_{\mathcal{D}} \to \xi$ via our contribution: the experience-driven random trees planners ERT and ERTConnect presented in Section III. We empirically demonstrate that, when using our planner, a unique prior path suffices to solve other instances of the same task.

III. EXPERIENCE-DRIVEN RANDOM TREES

The ERT and ERTConnect planners are inspired by tree sampling-based methods [18], [19]. Our planners, however, iteratively leverage a single task-relevant prior path experience by parts (segments, a.k.a., micro-experiences) to ease the capture of connectivity of the space. Such micro-experiences are semi-randomly morphed to generate task-relevant motions, i.e., segments that resemble those in the prior (e.g., dotted lines in Figure 2). The obtained motions are sequentially concatenated to compose a task-relevant tree (see green tree in Figure 3). This exploratory strategy aims at finding a trace

along the tree edges, i.e., a sequence of local modifications on the prior, that constitutes a continuous path ξ which satisfies $\xi : \alpha \in [0,1] \rightarrow \mathcal{S}_{\text{free}}$, $\xi(0) = \mathbf{q}_{\text{start}}$ and $\xi(1) = \mathbf{q}_{\text{goal}}$.

Noteworthy, our planners are designed to be agnostic to distance metrics, as capturing proximity between two robot configurations in a task is not trivial. Moreover, such metric would potentially need to be designed for each task. Therefore, instead of iteratively growing a tree from the nearest configuration to a random sample (RRT-like [19]), our experience-driven random trees iteratively branch-off (expand, EST-like [18]) by concatenating the inferred motions. Likewise, to generate resembling motions, we deform the micro-experiences such that no similarity metric is needed.

The core routine through which the planner exploits the prior experience to generate task-relevant micro-experiences is detailed in Section III-A, and its usage in a uni- and bi-directional sampling-based planning strategy is presented in Section III-B and Section III-C, respectively.

A. Inferring Task-relevant Motions from a Single Experience

For planning efficiently, we are particularly interested in generating motions that are task-relevant in $\mathcal{S}$, i.e., coherent according to the robot state in a task. To that purpose, our planners leverage an experience by parts in a "malleable" fashion, as opposed to the common "rigid" usage, to infer motions that are likely to be relevant to different task instantiations.

Initially, our planners pre-process the given experience $\xi_{\mathcal{D}}$ before exploiting it iteratively. Specifically, $\xi_{\mathcal{D}}$ is mapped onto the current planning problem to obtain $\xi'_{\mathcal{D}}$, a path whose initial and final configurations match the start and goal of the current planning problem (see Figure 2). The computation of such mapping $\xi_{\mathcal{D}} \rightarrow \xi'_{\mathcal{D}}$ is detailed within the description of the planners. Then, at each iteration, our planners leverage a part (micro-experience) of the mapped experience $\xi'_{\mathcal{D}}$ to infer suitable motions for the task. Generally, let $\psi_{\mathcal{D}} : \alpha \in [\alpha_{\text{ini}}, \alpha_{\text{end}}]$ be a micro-experience from the prior spanning from α_{ini} to α_{end} such that $\psi_{\mathcal{D}}(\alpha) = \xi'_{\mathcal{D}}(\alpha)\ \forall\ \alpha \in [\alpha_{\text{ini}}, \alpha_{\text{end}}]$ (e.g., red segment in Figure 2). We denote the extraction of a micro-experience from a prior as $\psi_{\mathcal{D}} = \xi'_{\mathcal{D}}(\alpha_{\text{ini}}, \alpha_{\text{end}})$, and say that such segment has a phase span $|\psi_{\mathcal{D}\alpha}| \in (0,1]$.

Extracted micro-experiences are exploited to create task-relevant motions. Formally, let $\nu : \psi_{\mathcal{D}} \rightarrow \psi \in \mathbb{R}^{(n+1)\times(n+1)}$ be a function that morphs a sequence of states onto another region of $\mathcal{S}$. We formulate the support of this operation to be that of an affine transformation of the form $\psi = A\psi_{\mathcal{D}} + B$, where $\psi_{\mathcal{D}(n+1)\times k} = \langle \bar{\mathbf{q}}_{\mathcal{D}n\times k}, \bar{\alpha}_{\mathcal{D}1\times k} \rangle$ is a prior micro-experience with k states, and $\psi_{(n+1)\times k} = \langle \bar{\mathbf{q}}_{n\times k}, \bar{\alpha}_{1\times k} \rangle$ is the generated task-relevant segment. Specifically, we design A to be a shear transform for its shape-preserving properties, and B to be a translation of the segment into a region of interest. Formally, then, this affine transformation modulates $\psi_{\mathcal{D}}$ as:

$$\begin{bmatrix} \bar{\mathbf{q}} \\ \bar{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbb{I}_{n\times n} & \boldsymbol{\lambda}_{n\times 1} \\ \mathbb{0}_{1\times n} & |\psi_{\mathcal{D}\alpha}| \end{bmatrix} \begin{bmatrix} \bar{\mathbf{q}}_{\mathcal{D}} \\ \boldsymbol{\rho} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_{n\times 1} & \dots & \mathbf{b}_{n\times 1} \\ \alpha_{\text{ini}} & \dots & \alpha_{\text{ini}} \end{bmatrix}_{(n+1)\times k}, \quad (2)$$

where $\boldsymbol{\lambda}_{n\times 1}$ is the shearing coefficient, $\mathbf{b}_{n\times 1}$ is a shifting vector, and $\boldsymbol{\rho} = [0, ..., 1]_{1\times k}$ is a local reparametrisation of $\bar{\alpha}_{\mathcal{D}}$. Note that the phase of the generated segment ψ remains $\bar{\alpha} = \bar{\alpha}_{\mathcal{D}}$. Informally, Equation 2 translates and smoothly deforms the micro-experience by adding up the increasing amount of noise $\boldsymbol{\lambda}\boldsymbol{\rho} + \mathbf{b}$, such that $\psi(\alpha_{\text{ini}}) - \psi_{\mathcal{D}}(\alpha_{\text{ini}}) = \mathbf{b}$ and $\psi(\alpha_{\text{end}}) - \psi_{\mathcal{D}}(\alpha_{\text{end}}) = \boldsymbol{\lambda} + \mathbf{b}$. Therefore, specifying $\mathbf{b}$ and $\boldsymbol{\lambda}$ enables the generation of new micro-experiences and their mapping onto any region of interest in $\mathcal{S}$. Figure 2 exemplifies the affine morphing in Equation 2 with different parameters. We detail the implementation of Equation 2 in Algorithm 1.

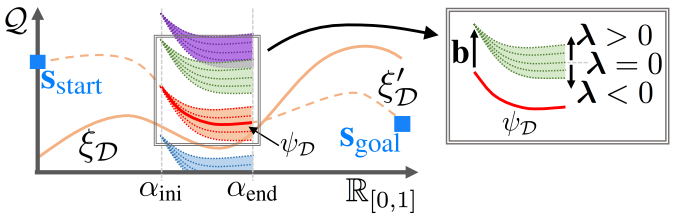


Fig. 2: Illustrative example of Equation 2: generation of resembling motions (dotted lines) by morphing the micro-experience $\psi_{\mathcal{D}}$ with semi-random $\mathbf{b}$ (shift) and $\boldsymbol{\lambda}$ (shear) pairs.

Algorithm 1: MORPH_SEGMENT($\psi_{\mathcal{D}}$, $\boldsymbol{\lambda}$, $\mathbf{b}$)

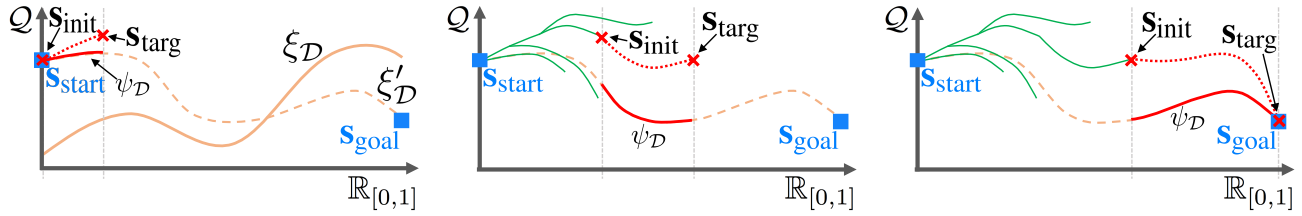
Input:
$\psi_{\mathcal{D}}$: micro-experience of phase-span $|\psi_{\mathcal{D}\alpha}|$ from α_{ini}
$\boldsymbol{\lambda}$: shearing coefficient
$\mathbf{b}$: shifting vector
Output:
ψ: morphed motion

1 **for** $\rho \leftarrow 0$ **to** 1 **do** `// Equation 2`
2 $\quad \alpha = \rho|\psi_{\mathcal{D}\alpha}| + \alpha_{\text{ini}}$
3 $\quad \psi(\alpha) \leftarrow \psi_{\mathcal{D}}(\alpha) + \rho\boldsymbol{\lambda} + \mathbf{b}$
4 **return** ψ

We exploit the ability to morph parts of the mapped prior path experience $\xi'_{\mathcal{D}}$ to infer motions that are suitable to either *connect* two particular states $\mathbf{s}_{\text{init}}$ and $\mathbf{s}_{\text{targ}}$, or *explore* the best way to continue the task from a given state $\mathbf{s}_{\text{init}}$. These two processes, and their interaction with Algorithm 1, are detailed in Algorithm 2 and illustrated in Figure 3.

Connect (line 3 to 6): given two task-related configuration-phase states $\mathbf{s}_{\text{init}} = \langle \mathbf{q}_{\text{init}}, \alpha_{\text{init}} \rangle$ and $\mathbf{s}_{\text{targ}} = \langle \mathbf{q}_{\text{targ}}, \alpha_{\text{targ}} \rangle$, we hypothesise that a suitable connection may result from mapping the micro-experience $\psi_{\mathcal{D}} : \alpha \in [\alpha_{\text{init}}, \alpha_{\text{targ}}]$ between $\mathbf{s}_{\text{init}}$ and $\mathbf{s}_{\text{targ}}$. Thus, after extracting the relevant micro-experience from $\xi'_{\mathcal{D}}$ (line 4), the parameters $\mathbf{b}$ and $\boldsymbol{\lambda}$ of the mapping in Equation 2 are calculated such that the resulting micro-experience ψ satisfies $\psi(\alpha_{\text{init}}) = \mathbf{q}_{\text{init}}$ and $\psi(\alpha_{\text{targ}}) = \mathbf{q}_{\text{targ}}$ (line 5 and 6).

Explore (line 8 to 11): given one task-related configuration-phase state $\mathbf{s}_{\text{init}} = \langle \mathbf{q}_{\text{init}}, \alpha_{\text{init}} \rangle$, we hypothesise that a suitable continuation of the task is to apply a micro-experience similar to that $\psi_{\mathcal{D}} \in \xi'_{\mathcal{D}}$ starting at α_{init}. For that, we first determine which span of $\xi'_{\mathcal{D}}$ to exploit by defining α_{targ} (line 8). An appropriate α_{targ} depends on the direction in which $\xi'_{\mathcal{D}}$ is being exploited; we call it `forward` when exploiting the prior from $\xi'_{\mathcal{D}}(0)$ to $\xi'_{\mathcal{D}}(1)$, and `backward` otherwise. Correspondingly,



(a) ERT: first iteration example (b) ERT: *explore* example (c) ERT: *connect* example

Fig. 3: Experience-driven random trees iteratively build a tree (green) of micro-experiences. At each iteration, an existing node in the tree is randomly selected to either *explore* the most suitable continuation of the task (e.g., snapshots in (a) and (b)), or *connect* to another known state (e.g., the goal state as in (c) (ERT), or a state in the other tree (ERTConnect)). In both cases, relevant motions (dotted red) are generated by morphing micro-experiences (red) of the prior path experience $\xi'_{\mathcal{D}}$ (see Figure 2).

SAMPLE_SEGMENT_END(α_{init}) defines α_{targ} as:

$$\alpha_{\text{targ}} = \begin{cases} \min(\alpha_{\text{init}} + \mathbb{U}(\omega_{\min},\ \omega_{\max}),\ 1), & \text{if } \texttt{forward} \\ \max(0,\ \alpha_{\text{init}} - \mathbb{U}(\omega_{\min},\ \omega_{\max})), & \text{if } \texttt{backward} \end{cases} \tag{3}$$

where $\mathbb{U}(\omega_{\min}, \omega_{\max})$ draws a sample from a uniform distribution to determine the phase span of the extracted segment. The bounds $\omega_{\min}$ and $\omega_{\max}$ are discussed in Section IV-B. Next, the corresponding segment $\psi_{\mathcal{D}} : \alpha \in [\alpha_{\text{init}}, \alpha_{\text{targ}}]$ is extracted from the mapped prior path experience $\xi'_{\mathcal{D}}$ (line 9), and $\mathbf{b}$ is computed for the resulting micro-experience ψ to start at $\mathbf{s}_{\text{init}}$, i.e., to satisfy $\psi(\alpha_{\text{init}}) = \mathbf{q}_{\text{init}}$ (line 10). Finally, to generate a task-relevant motion from $\mathbf{q}_{\text{init}}$, the shearing coefficient $\boldsymbol{\lambda}$ is sampled randomly within some bounds to morph the micro-experience $\psi_{\mathcal{D}}$ into a similar motion (line 11). In particular, $\boldsymbol{\lambda}$ is drawn from a uniform distribution $\mathbb{U}(-\boldsymbol{\epsilon}|\psi_{\mathcal{D}_\alpha}|,\ \boldsymbol{\epsilon}|\psi_{\mathcal{D}_\alpha}|)$ such that, at each iteration, the maximum allowed deformation is proportional to the segment's phase span. This implies that the accumulated deformation along any possible path ξ found by our planners does not exceed, with respect to $\xi'_{\mathcal{D}}$, the user-defined malleability bound $\boldsymbol{\epsilon}$ (see discussion in Section IV-B).

Overall, the method GENERATE_SEGMENT($\cdot$) enables the presented experience-guided random tree planners to leverage a single path experience at different levels of granularity, and map task-relevant segments onto any region of interest in the configuration-phase space. In that way, our planner aims at composing a valid path from a suitable sequence of morphed micro-experiences. The remaining of this section discusses the usage of such routine in our uni-directional (ERT) and a bi-directional (ERTConnect) tree sampling-based techniques.

Algorithm 2: GENERATE_SEGMENT($\mathbf{s}_{\text{init}}$, $\mathbf{s}_{\text{targ}}$, $\xi'_{\mathcal{D}}$)

Input:
$\mathbf{s}_{\text{init}}$: required segment configuration-phase start
$\mathbf{s}_{\text{targ}}$: required (if any) segment configuration-phase end
$\xi'_{\mathcal{D}}$: prior experience
Output:
ψ: generated segment
$\mathbf{s}_{\text{end}}$: end configuration-phase of the segment ψ

1 $\langle \mathbf{q}_{\text{init}}, \alpha_{\text{init}} \rangle = \mathbf{s}_{\text{init}}$
2 **if not** $s_{targ} = \varnothing$ **then** // connect
3 $\quad \langle \mathbf{q}_{\text{targ}}, \alpha_{\text{targ}} \rangle = \mathbf{s}_{\text{targ}}$
4 $\quad \psi_{\mathcal{D}} \leftarrow \xi'_{\mathcal{D}}(\alpha_{\text{init}},\ \alpha_{\text{targ}})$
5 $\quad \mathbf{b} \leftarrow \mathbf{q}_{\text{init}} - \psi_{\mathcal{D}}(\alpha_{\text{init}})$
6 $\quad \boldsymbol{\lambda} \leftarrow \mathbf{q}_{\text{targ}} - (\psi_{\mathcal{D}}(\alpha_{\text{targ}}) + \mathbf{b})$
7 **else** // explore
8 $\quad \alpha_{\text{targ}} \leftarrow$ SAMPLE_SEGMENT_END(α_{init})
9 $\quad \psi_{\mathcal{D}} \leftarrow \xi'_{\mathcal{D}}(\alpha_{\text{init}},\ \alpha_{\text{targ}})$
10 $\quad \mathbf{b} \leftarrow \mathbf{q}_{\text{init}} - \psi_{\mathcal{D}}(\alpha_{\text{init}})$
11 $\quad \boldsymbol{\lambda} \leftarrow \mathbb{U}(-\boldsymbol{\epsilon}|\psi_{\mathcal{D}_\alpha}|,\ \boldsymbol{\epsilon}|\psi_{\mathcal{D}_\alpha}|)$
12 $\psi \leftarrow$ MORPH_SEGMENT($\psi_{\mathcal{D}}$, $\boldsymbol{\lambda}$, $\mathbf{b}$)
13 $\mathbf{s}_{\text{end}} \leftarrow \langle \psi(\alpha_{\text{targ}}), \alpha_{\text{targ}} \rangle$
14 **return** $\langle \psi,\ \mathbf{s}_{\text{end}} \rangle$

B. Uni-directional Experience-driven Random Trees (ERT)

Algorithm 3 provides the pseudo-code of the uni-directional version of our planner. The algorithm seeks finding a continu-

Algorithm 3: ERT($\mathbf{s}_{\text{start}}$, $\mathbf{s}_{\text{goal}}$, $\xi_{\mathcal{D}}$)

Input:
$\mathbf{s}_{\text{start}}$ and $\mathbf{s}_{\text{goal}}$: start and goal configuration-phase
$\xi_{\mathcal{D}}$: prior experience
Output:
ξ: collision-free path

/* map $\xi_{\mathcal{D}}$ onto current problem */
1 $\langle \xi'_{\mathcal{D}}, \varnothing \rangle \leftarrow$ GENERATE_SEGMENT($\mathbf{s}_{\text{start}}$, $\mathbf{s}_{\text{goal}}$, $\xi_{\mathcal{D}}$)
2 **if** IS_VALID($\xi'_{\mathcal{D}}$) **then**
3 $\quad$ **return** $\xi'_{\mathcal{D}}$

/* sampling-based $\xi'_{\mathcal{D}}$ exploitation */
4 $\mathcal{T}$.init($\mathbf{s}_{\text{start}}$)
5 **while not** STOPPING_CONDITION() **do**
$\quad$ /* node selection */
6 $\quad \mathbf{s}_{\text{init}} \leftarrow \mathcal{T}$.select_node()
$\quad$ /* micro-experience generation */
7 $\quad \mathbf{s}_{\text{targ}} \leftarrow \varnothing$
8 $\quad$ **if** ATTEMPT_GOAL() = *True* **then**
9 $\quad\quad \mathbf{s}_{\text{targ}} \leftarrow \mathbf{s}_{\text{goal}}$
10 $\quad \langle \psi, \mathbf{s}_{\text{targ}} \rangle \leftarrow$ GENERATE_SEGMENT($\mathbf{s}_{\text{init}}$, $\mathbf{s}_{\text{targ}}$, $\xi'_{\mathcal{D}}$)
$\quad$ /* tree extension */
11 $\quad$ **if** EXTEND($\mathcal{T}$, ψ, $\mathbf{s}_{\text{init}}$, $\mathbf{s}_{\text{targ}}$) $\neq$ *Failed* **then**
12 $\quad\quad$ **if** GOAL_REACHED($\mathbf{s}_{\text{targ}}$) **then**
13 $\quad\quad\quad$ **return** PATH($\mathcal{T}$)

Algorithm 4: EXTEND($\mathcal{T}$, ψ, $\mathbf{s}_{\text{init}}$, $\mathbf{s}_{\text{targ}}$)

Input:
$\mathcal{T}$: tree of previously generated micro-experiences
ψ: new generated micro-experience
$\mathbf{s}_{\text{init}}$ and $\mathbf{s}_{\text{targ}}$: start and end configuration-phase of ψ
Output:
outcome of the tree extension attempt

1 **if** IS_VALID(ψ) **then**
2 $\mathcal{T}$.add_vertex($\mathbf{s}_{\text{targ}}$)
3 $\mathcal{T}$.add_edge(ψ, $\mathbf{s}_{\text{init}}$, $\mathbf{s}_{\text{targ}}$)
4 **return** Advanced
5 **return** Failed

ous path from a start $\mathbf{s}_{\text{start}}$ to a goal $\mathbf{s}_{\text{goal}}$ configuration, given a related path experience $\xi_{\mathcal{D}}$. The planner firstly maps the entire prior experience $\nu : \xi_{\mathcal{D}} \rightarrow \xi'_{\mathcal{D}}$ onto the current planning problem (line 1); note that the output of GENERATE_SEGMENT($\cdot$) (Algorithm 2) is a segment ψ that spans from $\alpha_{\text{ini}} = 0$ to $\alpha_{\text{end}} = 1$, thus we rename it $\xi'_{\mathcal{D}}$. If $\xi'_{\mathcal{D}}$ is not valid (line 2), the planner proceeds to exploit $\xi'_{\mathcal{D}}$ to generate task-relevant micro-experiences. The planner follows a three-step procedure (node selection, segment sampling, and tree extension) until the stopping condition is met (line 5). A node $\mathbf{s}_{\text{init}}$ is selected from the tree $\mathcal{T}$ with probability $P(node) = \frac{1}{w(node)+1}$ (line 6), where $w(\cdot)$ is a weighting function that penalises the selection of a node according to the number of times that it has already been selected. This weighted sampling strategy seeks a uniform selection of all nodes over time, thus promoting a first depth exploration of the task phase α. From the selected node $\mathbf{s}_{\text{init}}$, the tree is expanded using segments that resemble those in the prior experience $\xi'_{\mathcal{D}}$. With probability p, the expansion of the tree attempts to *connect* $\mathbf{s}_{\text{init}}$ with $\mathbf{s}_{\text{goal}}$, whereas with probability $(1 - p)$ an *explore* expansion is done towards a semi-random configuration $\mathbf{s}_{\text{targ}}$ (line 7 to 10). Algorithm 2, previously explained in Section III-A, details the extraction of suitable segments under these two different cases. The extracted segment is used to attempt expanding the tree (line 11) following Algorithm 4. If the segment is valid, it is integrated into the tree. Note that, as discussed in Section III-A, the appended segment is a motion whose shape resembles that of the related micro-experiences in the prior experience, not a straight line. Finally, if the incorporated (valid) segment reaches the goal, the path is returned (line 12 and 13).

C. Bi-directional ERT (ERTConnect)

The principles of leveraging from a prior experience by generation of task-relevant micro-experiences can also be employed in a bi-directional fashion. The proposed bi-directional planning scheme resembles, in spirit, that of the RRTConnect [19], i.e., to simultaneously grow two trees, one from the start configuration and the other from the goal configuration, aiming to find a solution by connecting both trees. ERTConnect, however, includes the peculiarities of our experience-based planning approach. As shown in Algorithm 5, the planner firstly maps the prior path experience $\nu : \xi_{\mathcal{D}} \rightarrow \xi'_{\mathcal{D}}$ onto the current planning problem (line 1). If $\xi'_{\mathcal{D}}$ is not valid

Algorithm 5: ERTConnect($\mathbf{s}_{\text{start}}$, $\mathbf{s}_{\text{goal}}$, $\xi_{\mathcal{D}}$)

Input:
$\mathbf{s}_{\text{start}}$ and $\mathbf{s}_{\text{goal}}$: start and goal configuration-phase
$\xi_{\mathcal{D}}$: prior experience
Output:
ξ: collision-free path

/* map $\xi_{\mathcal{D}}$ onto current problem */
1 $\langle \xi'_{\mathcal{D}}, \varnothing \rangle \leftarrow$ GENERATE_SEGMENT($\mathbf{s}_{\text{start}}$, $\mathbf{s}_{\text{goal}}$, $\xi_{\mathcal{D}}$)
2 **if** IS_VALID($\xi'_{\mathcal{D}}$) **then**
3 **return** $\xi'_{\mathcal{D}}$

/* sampling-based $\xi'_{\mathcal{D}}$ exploitation */
4 $\mathcal{T}_a$.init($\mathbf{s}_{\text{start}}$)
5 $\mathcal{T}_b$.init($\mathbf{s}_{\text{goal}}$)
6 **while not** STOPPING_CONDITION() **do**
 /* node selection */
7 $\mathbf{s}_{\text{init}} \leftarrow \mathcal{T}_a$.select_node()
 /* micro-experience generation */
8 $\langle \psi, \mathbf{s}_{\text{targ}} \rangle \leftarrow$ GENERATE_SEGMENT($\mathbf{s}_{\text{init}}$, $\varnothing$, $\xi'_{\mathcal{D}}$)
 /* tree extension */
9 **if** EXTEND($\mathcal{T}_a$, ψ, $\mathbf{s}_{\text{init}}$, $\mathbf{s}_{\text{targ}}$) $\neq$ *Failed* **then**
10 **if** OTHER_EXTREME_REACHED($\mathbf{s}_{\text{targ}}$) **then**
11 **return** PATH($\mathcal{T}_a$)
 /* micro-experience generation */
12 $\mathbf{s}_{\text{near}} \leftarrow \mathcal{T}_b$.nearest_neighbour($\mathbf{s}_{\text{targ}}$)
13 $\langle \psi, \mathbf{s}_{\text{targ}} \rangle \leftarrow$ GENERATE_SEGMENT($\mathbf{s}_{\text{near}}$, $\mathbf{s}_{\text{targ}}$, $\xi'_{\mathcal{D}}$)
 /* tree connection */
14 **if** EXTEND($\mathcal{T}_b$, ψ, $\mathbf{s}_{\text{near}}$, $\mathbf{s}_{\text{targ}}$) $\neq$ *Failed* **then**
15 **return** PATH($\mathcal{T}_a$, $\mathcal{T}_b$)
16 SWAP($\mathcal{T}_a$, $\mathcal{T}_b$)

(line 2), the planner proceeds to exploit $\xi'_{\mathcal{D}}$ to compute a solution. The planner simultaneously grows two trees, one rooted at $\mathbf{s}_{\text{start}}$ and the other at $\mathbf{s}_{\text{goal}}$ (line 4 and 5). At each iteration, until the stopping criterion is met (line 6), a node of the active tree $\mathcal{T}_a$ is selected via weighted selection (line 7) to *explore* the space via a task-relevant micro-experience (line 8). If the active tree is extended successfully with the generated segment (line 9), we first check whether the end of the motion has reached the other extreme (line 10). This implies that a path has been found before the trees connected, either by $\mathcal{T}_a$ reaching the root of $\mathcal{T}_b$ or the other way around, in which case the path is returned as a solution (line 11). Otherwise, the node of $\mathcal{T}_b$ nearest to $\mathbf{s}_{\text{targ}}$ is selected (line 12) to attempt to *connect* both trees with a task-relevant segment (line 13). For such nearest neighbour query, we consider the Euclidean distance between the configuration components (without the phase). If the extension is successful, the corresponding path is returned (line 14 and 15).

IV. USING ERT AND ERTCONNECT

In this section we discuss some details on using the proposed experience-driven random trees planners.

A. Selecting a Prior from a Library of Experiences

A robot might have at its disposal a library $\mathcal{A}$ of task-relevant path experiences. As our planners exploit a unique prior path experience $\xi_{\mathcal{D}}$ to solve other instances of the same task, our current selection criteria $\xi_{\mathcal{D}} \in \mathcal{A}$ is to pick the prior that resembles the current planning query the most. Intuitively, if a solution to the current planning problem lies in the neighbourhood of a prior experience, the invariant robotic constraints encoded in the experience itself, such as self-collisions and joint limits, are more likely to prevail and thus, ease the planner's computations. Inspired by the experience selection in [11], we estimate such resemblance by ranking the experiences for their similarity to the start and goal of the planning query. This is, the prior experience $\xi_{\mathcal{D}}$ selected to feed the proposed experience-based planner is such that:

$$\xi_{\mathcal{D}} = \arg\min_{\xi_{\mathcal{D}i} \in \mathcal{A}} \text{dist}(\xi_{\mathcal{D}i}(0),\ \mathbf{q}_{\text{start}}) + \text{dist}(\xi_{\mathcal{D}i}(1),\ \mathbf{q}_{\text{goal}}), \quad (4)$$

where $\mathbf{q}_{\text{start}}$ and $\mathbf{q}_{\text{goal}}$ are the start and goal configurations of the current planning problem, and $\text{dist}(\cdot)$ is a function that estimates the Euclidean distance between configurations in $\mathcal{Q}$.

We verified the approach to select a unique prior from a library of experiences described in Equation 4 experimentally; despite it led to good results, the topic merits further attention.

B. Planner's Parameterisation

Next, we review the parameters of the presented planners:

- p - probability of attempting to *connect* the tree to the goal (only in ERT). This parameter should be set small to allow the planner *explore* the space. Default: $p = 0.05$.
- ω_{min} and ω_{max} - lower and upper phase span bounds of the extracted segments. Indirectly, these parameters delimit the length of the motions added in the tree to *explore* the space. Default: $\omega_{\text{min}} = 0.05$ and $\omega_{\text{max}} = 0.1$.
- $\boldsymbol{\epsilon}$ - malleability bound to delimit the amount of morphing applied to the micro-experiences. Intuitively, this parameter defines a volume (tube) around $\xi'_{\mathcal{D}}$ where the planner can *explore* for a solution. Default: $\boldsymbol{\epsilon} = \mathbf{5}_{1\times n}$ (large enough to cover the entire robot's kinematic range).

Our planners' default parameters are non-optimised for a particular planning problem, but left generic to succeed in many scenarios provided a relevant path experience. The planners' behaviour can be adjusted by tuning, namely, ω_{min}, ω_{max} and $\boldsymbol{\epsilon}$. As our planner discards motions that are not entirely valid, large phase spans might endanger the ability to build a tree. However, in scenarios with few obstacles, ω_{min} and ω_{max} can be set large to speed up planning computations. Also, lowering $\boldsymbol{\epsilon}$ can speed up computations, as the tree growth would be more guided around the mapped experience $\xi'_{\mathcal{D}}$. The lower $\boldsymbol{\epsilon}$, the more dependant the planner is on the suitability of the provided experience as the probabilistic completeness is compromised. Knowing the level of dissimilarity between the experience and the current problem might aid in tuning $\boldsymbol{\epsilon}$ to trade growth guidance and space exploration.

V. EXPERIMENTAL EVALUATION

The proposed experience-guided random trees have been implemented in the Open Motion Planning Library (OMPL) [20] and evaluated on the Fetch robot [21] in a shelf-stocking task. Fetch is a humanoid robot with a 7-DoF arm attached on a sliding torso, thus requiring to plan in an 8-DoF configuration space. Our experiments are designed to measure the generalisation capabilities of our planner in scenarios that involve different levels of dissimilarity between prior experiences and task instances (see Section V-A). The considered task instances include synthetic and real-world scenarios (see Section V-B).

A. Experimental Setup

Our experimental setup considers a varied instance set of the challenging problem of reaching a target object in a shelving unit, specifically in a synthetic 4-tier (see Figure 4) and a narrower real 5-tier shelving unit (see Figure 5). Task instances in these scenarios not only present variability on the location of the shelving unit ($\pm 90^\circ$ around the robot) and the robot's initial position ($\pm$10cm), but also on the location of the target object and the obstacles within the shelving unit.

To further evaluate the generalisation capabilities of the proposed experience-based planner, we introduce some additional variability across the experimental setup. This is, we compute with RRTConnect a total of 100 experiences from different task instances; them all at the synthetic shelving unit, with target objects in the middle shelf and no obstacles. These scenarios are discarded for the rest of the evaluation. Then, these experiences are used to evaluate the planner in four scenario sets that involve increasing dissimilarity levels between experiences and planning queries:

- *Set 1*: 200 instances with target objects in the middle shelf without obstacles (synthetic). Note that these instances resemble those used to compute experiences.
- *Set 2*: 200 instances with target objects in the middle shelf with the presence of obstacles (synthetic).
- *Set 3*: 200 instances with target objects in three different shelves with the presence of obstacles (synthetic).
- *Set 4*: 120 instances with target objects and obstacles in the middle shelf (real-world).

The four sets of task instances are used to benchmark our bi-directional ERTConnect planner against RRTConnect [19] and the most representative experience-based planners that employ motions as prior information of the task, i.e., Thunder [14] and Lightning [11]. Note that these two frameworks are double-threaded with a bi-directional 'retrieve and repair' (RR) and 'plan from scratch' (PFS) module. Similarly, for a fair comparison, we embed our ERTConnect in a double-threaded framework which runs RRTConnect in parallel to PFS. The reported results indicate the contribution of the RR (plain bar) and PFS (stripped bar) modules in solving the planning queries separately, and the required planning time jointly (plain bar).

In this work's context, where we consider novel tasks instances in varied scenarios, optimising each planner's parameters across queries is not possible. Optimal parametrisation requires extensive testing in each scenario set, and thus knowing the scenarios in advance, among other planning aspects. Therefore, all planners are used in their default OMPL settings, and ours is set to the non-optimised default parameters

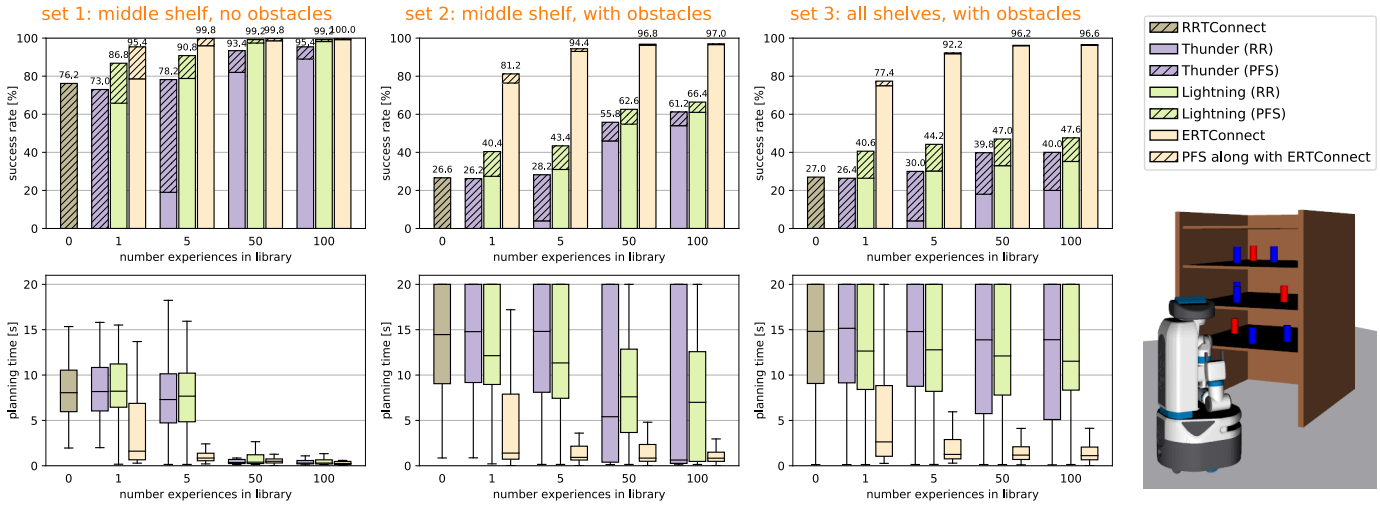


Fig. 4: Success rate and solving time results for the benchmark on synthetic scenarios, where the Fetch Robot needs to reach a target object in the shelving unit subject to multiple variations of the task instances. From left to middle-right column, case studies from less to more experience-instance dissimilarity: *Set 1*, *Set 2* and *Set 3*. The picture on the right depicts a particular instance of *Set 3* which, differently from the considered prior experiences, involves target objects (red cylinders) located at any of the three shelves, a different relative location of the shelving unit, as well as obstacles (blue cylinders).

specified in Section IV-B. The benchmark is run on an Intel i7 Linux machine with 4 3.6GHz cores and 16GB of RAM. The performance of the three experience-based planners in each instance set is assessed under libraries with $\{1, 5, 50, 100\}$ prior experiences. The experiences provided to each planner are the same. Each query is repeated 50 times with a planning timeout of 20 seconds. All in all, the conducted benchmark involves a total of $468{,}000$ planning queries.

B. Results on Synthetic and Real-world Scenarios

The results of the benchmark on *Set 1*, *Set 2* and *Set 3* are summarised in Figure 4, whilst those in the real-world *Set 4* are depicted in Figure 5. As it can be observed, provided a high number of experiences that are close to the current planning problem (i.e., *Set 1* with the library of 100 experiences), all planners achieve a high success rate with solving time of the order of milliseconds. This behaviour is expected as, given the experience-query similarity and the library size, it is likely that there exists a prior experience that nearly resembles the current query, thus involving minimum repairing.

As the dissimilarity between experiences and queries increases, experience-based planners need to generalise the prior information more broadly to succeed. Intuitively, the need of generalisation arises when a reduced number of demonstrations in the library needs to cover varied task instances (x-axis within each experimental set), or when the current planning requirements differ significantly from the set of available experiences (variability across experimental sets). The outcome of our benchmark points out that the performance of Thunder's RR module drops abruptly by either dissimilarity factor, whereas the Lighting's RR is not as affected by the lack of experiences as it is when dealing with significantly different task instances. The poor generalisation of these frameworks across instances is due to the rigid usage of prior experiences. Our approach, instead, by leveraging experiences in a malleable way, achieves a success rate and solving time that significantly improves that of Lightning and Thunder.

The importance of generalising prior information is particularly noticeable in the real-world task instances in *Set 4*, where the queries differ from the experiences not only on the location of the shelving unit and the target object, but also on the narrower geometry of the whole shelving unit, the height of the shelf where the target object is located at, and the presence of obstacles. Under these challenging task variations and when accounting with only one demonstration, our approach outperforms by a factor of approximately 3.7

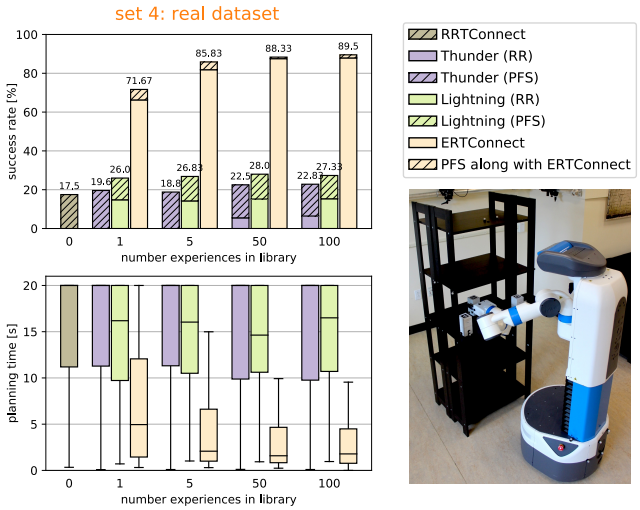


Fig. 5: Success rate and solving time results for the benchmark on real-world scenarios (*Set 4*), where the Fetch Robot needs to reach a target object by generalising prior experiences to a narrower shelving unit geometry, to different locations of the shelving unit, robot's initial position and target object, as well as to the presence of obstacles.

and 37.1 times the RR module of the Lightning and Thunder frameworks, respectively. Similarly, those frameworks are respectively outperformed by our approach by a factor of 4.7 and 8.9 times when considering 100 experiences. Notably, while these planners time out most of the trials, our method required only a quarter (with one demonstration) and less than an eight (with multiple demonstrations) of the planning time budget to find a solution.

The ability to generalise prior information not only limits the level of experience-query dissimilarity that a planner can cope with, but also the number of experiences that are required to achieve high performance. As an example, providing 50 experiences to Thunder's RR, 5 to Lightning's RR and 1 to our ERTConnect leads to approximately the same success rate of 80% in *Set 1*. Achieving such a performance in the other experimental sets with Lightning and Thunder is not possible even with a library of 100 experiences, whereas our approach surpasses such performance when selecting a unique experience from a library of only 5 experiences. This implies that our ERTConnect planner, by generalising prior experiences more efficiently, significantly outperforms current experienced-based planners using libraries of motions in the literature even when provided with notably fewer experiences.

VI. DISCUSSION

In this manuscript, we have presented two new experience-based planners: the uni-directional experience-driven random tress (ERT) and the bi-directional ERT (ERTConenct). These two methods are tree sampling-based planners that iteratively exploit a single prior path experience to ease the capture of connectivity of the space. At each iteration, a segment of the prior is extracted and semi-randomly morphed to generate a task-relevant motion. The obtained motions are sequentially concatenated to compose a task-relevant tree, such that a trace along the edges constitutes a solution to a given task-related planning problem. Thorough experimentation against current experienced-based planners using libraries of motions in the literature [11], [14] demonstrates our planner's significant superior performance in a wide range of task instances.

We have shown that, similarly to related work [11], [14], our planner can be used in parallel with a planning from scratch strategy to guarantee probabilistic completeness. Therefore, when multi-threading is an option, a planning from scratch thread should be considered, as well as multiple instantiations of our planner with a set of varied experiences that maximises space coverage. In the future, we plan to explore the convenience of different transformation supports to infer relevant micro-experiences subject to intrinsic task and robot constraints; for instance, early tests demonstrate our planners' suitability to leverage experiences in $\mathrm{SO}(3)$ using quaternions. Another promising line for future work is the extension of our planner to leverage multiple experiences simultaneously, such that the local exploration is conducted with the most suitable segment in the library. Likewise, such a strategy would potentially allow the planner adapting to changes in the planning context, e.g., dynamic obstacles and moving goal configurations, as well as solving novel tasks by combining experiences of multiple different tasks.

ACKNOWLEDGMENTS

The authors thank Zachary Kingston for his support in integrating our planner into MoveIt, and Carlos Quintero for his help on the final experiments. Also, the authors are grateful to Paola Ardón for valuable discussions and suggestions.

REFERENCES

[1] F. Meier, D. Kappler, and S. Schaal, "Online learning of a memory for learning rates," in *2018 IEEE International Conference on Robotics and Automation*, pp. 2425–2432, IEEE, 2018.

[2] È. Pairet, P. Ardón, M. Mistry, and Y. Petillot, "Learning generalizable coupling terms for obstacle avoidance via low-dimensional geometric descriptors," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3979–3986, 2019.

[3] S. Stark, J. Peters, and E. Rueckert, "Experience reuse with probabilistic movement primitives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1210–1217, IEEE, 2019.

[4] È. Pairet, P. Ardón, M. Mistry, and Y. Petillot, "Learning and composing primitive skills for dual-arm manipulation," in *Annual Conference Towards Autonomous Robotic Systems*, pp. 65–77, Springer, 2019.

[5] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, 2020.

[6] M. Zucker, J. Kuffner, and J. A. Bagnell, "Adaptive workspace biasing for sampling-based planners," in *2008 IEEE International Conference on Robotics and Automation*, pp. 3757–3762, IEEE, 2008.

[7] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation*, pp. 7087–7094, IEEE, 2018.

[8] P. Lehner and A. Albu-Schäffer, "The repetition roadmap for repetitive constrained motion planning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3884–3891, 2018.

[9] C. Chamzas, A. Shrivastava, and L. E. Kavraki, "Using local experiences for global motion planning," in *2019 International Conference on Robotics and Automation*, pp. 8606–8612, IEEE, 2019.

[10] D. Molina, K. Kumar, and S. Srivastava, "Learn and link: learning critical regions for efficient planning," in *IEEE International Conference on Robotics and Automation*, 2020.

[11] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *2012 IEEE International Conference on Robotics and Automation*, pp. 3671–3678, IEEE, 2012.

[12] N. Jetchev and M. Toussaint, "Fast motion planning from experience: trajectory prediction for speeding up movement generation," *Autonomous Robots*, vol. 34, no. 1-2, pp. 111–127, 2013.

[13] M. Phillips, B. J. Cohen, S. Chitta, and M. Likhachev, "E-graphs: bootstrapping planning with experience graphs.," in *Robotics: Science and Systems*, 2012.

[14] D. Coleman, I. A. Şucan, M. Moll, K. Okada, and N. Correll, "Experience-based planning with sparse roadmap spanners," in *2015 IEEE International Conference on Robotics and Automation*, pp. 900–905, IEEE, 2015.

[15] Y. Wang, K. Harada, and W. Wan, "Motion planning through demonstration to deal with complex motions in assembly process," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots*, pp. 544–550, IEEE, 2019.

[16] J. DelPreto, J. I. Lipton, L. Sanneman, A. J. Fay, C. Fourie, C. Choi, and D. Rus, "Helping robots learn: a human-robot master-apprentice model using demonstrations via virtual reality teleoperation," in *IEEE International Conference on Robotics and Automation*, 2020.

[17] P. Ardón, È. Pairet, Y. Petillot, R. P. Petrick, S. Ramamoorthy, and K. S. Lohan, "Self-assessment of grasp affordance transfer," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2020.

[18] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings of International Conference on Robotics and Automation*, vol. 3, pp. 2719–2726, IEEE, 1997.

[19] J. J. Kuffner and S. M. LaValle, "RRT-connect: an efficient approach to single-query path planning," in *2000 International Conference on Robotics and Automation*, vol. 2, pp. 995–1001, IEEE, 2000.

[20] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, pp. 72–82, December 2012.

[21] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch and Freight: standard platforms for service robot applications," in *Workshop on autonomous mobile service robots*, 2016.