Experience-Based Path Planning for Mobile Robots Exploiting User Preferences

Lorenzo Nardi Cyrill Stachniss

Abstract— **The demand for flexible industrial robotic solutions that are able to accomplish tasks at different locations in a factory is growing more and more. When deploying mobile robots in a factory environment, the predictability and reproducibility of their behaviors become important and are often requested. In this paper, we propose an easy-to-use motion planning scheme that can take into account user preferences for robot navigation. The preferences are extracted implicitly from the previous experiences or from demonstrations and are automatically considered in the subsequent planning steps. This leads to reproducible and thus better to predict navigation behaviors of the robot, without requiring experts to hard-coding control strategies or cost functions within a planner. Our system has been implemented and evaluated on a simulated KUKA mobile robot in different environments.**

I. Introduction

Nowadays, the use of robotic systems in manufacturing industries is widespread. In this context, robots are often fixed and employed for a specific task. Over the last decade, different industries started asking for more flexible robotic solutions able to accomplish different tasks such as navigation or mobile manipulation. Mobile robots operating on factory floors are usually not completely free to move in the whole environment. Often, users demand these robots to follow or prefer certain schemes for navigation. In such environments, robots are also frequently required to share the workspace with human workers. Especially in a shared workspace, the predictability and reproducibility of the robot behaviors are important.

In this paper, we investigate the problem of how to incorporate user preferences in robot navigation and reuse this knowledge when planning new trajectories. We aim at realizing a planning scheme that implicitly collects information through previous experiences. This is generally more convenient than hard-coding preferences. Formalizing preferences can be difficult and hard-coded rules are often complex to maintain and update. Furthermore, hard-coded rules may interfere and increase the complexity of switching between preferences or between different users. In contrast, even a non-expert user can take advantage from teaching robots through demonstrations, for example joysticking them along desired routes in the environment.

The contribution of this paper is a path planning system for robot navigation that aims at providing predictable and reproducible behaviors according to user preferences. We realize this by enabling the user to either demonstrate the robot a certain behavior or to provide feedbacks about experienced tasks. For each task the robot has accomplished, the user may rate the corresponding solution. If a solution is regarded as good, it is stored and reused to guide the planning process in a new but similar navigation task. We consider a situation descriptor for comparing tasks and an appropriate method for transferring experienced paths to new situations, similar to Jetchev and Toussaint [1]. We incorporate these two concepts into our planning system that is based on the idea of guided planning introduced by Jiang and Kallman [2]. Our system has been implemented within the Open Motion Planning Library (OMPL) framework [3] and using ROS.

Lorenzo Nardi and Cyrill Stachniss are with Institute for Geodesy and Geoinformation, University of Bonn, Germany. This work has partly been supported by the European Commission under the Horizon 2020 framework programme under grant agreement no. 645403 (RobDREAM).

II. Related Work

Motion planning for robot navigation has been intensively studied over the last decades. Typical motion planners start each search from scratch and seek to find a path to a given goal location. Due to their rapidity to discover the connectivity of a configuration space, sampling-based planners such as RRT-Connect [4] are frequently used for computing paths. When relying on a sampling-based planner, it is difficult for the user to make a prediction on how the resulting path will look like. In this work, we aim at finding a path in a similar similar manner as RRT but that is predictable according to the previous experiences of the robot.

Recently, several works on reusing experienced paths for motion planning have been proposed. Lien and Yu [5] construct roadmaps for obstacles and reuse them for similar obstacles during planning. Fraichard and Delsart [6] propose a scheme to deform previously computed trajectories in the configuration-time space. Phillips *et al.* [7] build an experience graph for representing the connectivity of the space required for the execution of repetitive tasks. The Lightning framework [8] retrieves paths from a database of previous generated trajectories and attempts to adapt them to the current planning problem. This idea is extended by Coleman *et al.* [9] that store generated paths in a sparse roadmap. Bruce and Veloso [10] introduced an approach for motion planning that extends the traditional RRT to reuse cached plans and biasing the search towards their waypoints. Zucker *et al.* [11] proposed a framework for adapting the sampling distribution to a problem class considering the features present in the underlying workspace. Jiang and Kallmann [2] presented the Attractor Guided Planner (*AGP*) to enable humanoid robots planning and executing tasks in dynamic environments. It improves the performance of sampling-based planners by storing every successful path and by biasing a new search to reproduce the structure of an experienced path according to a similarity function. If no similar task has been experienced or a path is not valid anymore due to changes in the environment, *AGP* changes back to a non-biased random search. The metric to identify similar task is rather straightforward and allows only for path queries in a single environment. Such approaches focus on taking advantage from previous experiences to speed up the computation time to find a path even in high-dimensional configuration spaces. Our work instead aims at exploiting previous trajectories in robot navigation to provide solutions that meet user expectations.

The problem of how to exploit data from previous experiences to generate appropriate trajectories in new situations has been addressed by Jetchev and Toussaint [1]. They optimize a cost function to learn the mapping between situations and trajectories. To do so, two main aspects are considered: the definition of an appropriate situation descriptor and an efficient task space transfer. The descriptor of the current situation is used to predict a trajectory that is transferred and optimized in this situation. This approach has been extended in [12], in which a voxel representation of the environment is used to generalize trajectories to a wider range of situations. We also use the concept of situation descriptor to predict paths for new situations, but we aim at generalizing over different environments and obstacles and learning the mapping between situations and experienced behaviors according to the preferences of the user of the system.

To achieve predictable trajectories, many researchers consider teach-and-repeat approaches for reliable robot autonomous navigation in which a robot seeks to repeat the same motion than during teaching. Sprunk *et al.* [13] consider trajectories that rely on scan matching to localize the robot relative to a taught trajectory and to achieve a highly accurate reproduction. Furgale *et al.* [14] propose an approach using a 3D spinning lidar that extends the standard teach-and-repeat by adding a local motion planner to account for dynamic environments. This is related to homing tasks addressed by Perea Ström *et al.* [15] for guiding a robot home in case the mapping system fails during an exploration mission. Mazuran *et al.* [16] propose to apply an optimization method to the demonstrated trajectories where the constraints are defined according to user preferences. In contrast to teach-and-repeat approaches, our objective is to maintain the flexibility of a real planner while considering user preferences.

Case-based reasoning is another approach related to our work. It considers robot experiences to build experimental models that are stored as cases and used in the future tasks. Meriçli *et al.* [17] adopted this approach to mobile push-manipulation, while Ros *et al.* [18] used it for action selection in the robot soccer domain.

In our system, we focus on mobile robot navigation and propose a path planning system based on a tailored version of *AGP* that allows for providing predictable behaviors even under changes in the environment or if the robot encounters an obstacle along its path. To do so, we define a situation descriptor and a path representation that allows for reusing experienced paths across different environments and obstacles. We exploit only the paths that have been deemed good by the user, meeting his preferences.

III. Approach

A. Use Case

We consider a setup in which a mobile robot is requested by an operator to perform navigation tasks in a given environment. This use case stems from the EU-funded H2020 project RobDREAM that focuses on an user-centered approach that allows non-experts to adapt the robot navigation behaviors and allows robots to improve over them.

In this scenario, the operator has the possibility to evaluate the robot behavior through a simple GUI rating a path as a *good* or a *bad* one. In addition to that, the operator can provide demonstrations to the robot as good examples by joysticking the platform. The good experiences are stored into a database. Over time, the database will grow and consist of those paths that have been approved by the user. Our work consists of a planning system that exploits such a database of examples to produce similar paths to those experienced. In this way, user preferences are taken into account without limiting the flexibility of a general planning system.

B. Overview of the System

Our system is realized through a two-level structure for robot navigation, in which both of the levels aim at generating paths that exploit the previous experiences. Given a start and goal robot configurations in an environment, the *global level* computes a plan between these two configurations. It depends on the global map and does not generalize the exploitation of experiences across different environments. The *local level* handles dynamic changes in the environment and plans deviations from the global path to avoid collisions with unforeseen obstacles while respecting the user preferences. To achieve this, only local obstacle locations matter, therefore it is largely independent from the environment itself and generalizes well to other scenes.

We consider three key concepts for realizing a flexible navigation system that can exploit positive experiences in the planning process. The remainder of this section *briefly* explains these concepts, which will be described in detail in the subsequent sections.

a) Path Representation: To describe a rated path or experience, we use an ordered list of relevant robot configurations along the path $A = \{a_1, \ldots, a_n\}$, called *attractors*. In a planning problem within the 2D plane, an attractor could be a (x, y, θ) configuration. An example of such attractors is given in Figure 1. Details are given in Section IV.

b) Situation Description: To plan exploiting similar previous experiences, it is essential to define a similarity relation among tasks. To this end, we introduce a *situation descriptor* to describe a task. A different description is used on the global and the local scale. The global descriptor d_g is

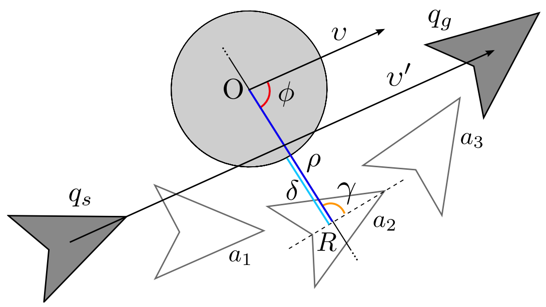


Fig. 1: Attractor representation of a local path $P = q_s, a_1, a_2, a_3, q_g$. A local attractor is described by (δ, ϕ, γ) with respect to the obstacle centered in O.

environment-dependent, while the local descriptor d_l relies just on the local setup. Further details are given in Section V.

c) Planning Exploiting Experiences: We propose a planning framework that employs a modified version of the Attractor Guided Planner (*AGP*) [2] on a global and a local planning level. The key idea is to use bi-directional RRT and to guide the sampling process to prefer trajectories that match previous experiences. If no previous experience matches the current situation, the system degrades to standard bi-directional RRT. As described in details in Section VI, the *global* and the *local* levels rely on different assumptions and run in parallel distinct planning instances.

IV. Path Representation

To represent a path, we recall the concept of *attractors* adopted in [2]. The attractors of a path are the set A of relevant robot configurations that allow for reproducing the structure of the corresponding path. Given a path P, it can be represented as $P = q_s, a_1, \ldots, a_n, q_g$, where $a_1, \ldots, a_n \in A$ and q_s and q_g are respectively the start and goal robot configurations. To save an experienced example, we extract its attractors and store them into the database.

In our system, we employ a window-based line fitting algorithm to compute the attractors of a path. It iteratively adds the path points to a window of dynamic size. If a line fits through the points in the window up to a threshold, the successive point is added. Otherwise, the last added point is identified as candidate attractor and the window is reinitialized. For each candidate attractor, we check whether a straight motion from the previous attractor is valid and collision free. If this is the case, the candidate attractor is added to the list of attractors for the corresponding path. Otherwise the previous point in the path is identified as new candidate attractor and a new check is performed. This procedures iterates until all path points are processed. The result is an ordered list of attractors represented in the environment frame describing the corresponding path.

The global and local level of our system differ for the way to store and reuse the attractors, so we maintain the examples in two distinct databases: D_G for the experienced global paths and D_L for the local ones. The global level depends on the global static map of the environment, therefore the attractors can be directly stored in map coordinates in D_G. The local level instead is not subordinate to the environment itself, so a transformation needs to be applied to the attractors to make them independent of the world frame. To this end, we consider a coordinate transformation to the coordinate system illustrated in Figure 1 that is based on the local path and current blocking obstacle. Our local coordinate system is defined by the vector v, that has the same orientation as the vector v' that connects the start to the goal positions, and is centered in the center of the obstacle O. In this frame, we identify each attractor by (ρ, ϕ, γ), where ρ is the distance of the robot to O, ϕ and γ are the angles that the line passing by O and the center of the robot R forms respectively with v and the robot axis. This coordinate system unambiguously determines the poses of the attractors considering only on local information, and accordingly allows to transfer experiences across different environments. To further generalize over different obstacles, instead of ρ, we can take into account δ, i.e. the distance of the attractor from the obstacle surface along ρ. Considering the coordinates (δ, ϕ, γ) allows to reuse an experienced local strategy even in the cases in which the obstacle encountered by the robot has different dimensions still keeping a safe distance from it. To exploit an experienced local path for a new navigation task, these local coordinates are transformed back in the frame of the current environment and used to guide the new planning.

V. Situation Description

To generalize properly from previous experiences when planning for a new task, it is fundamental to identify which of the experienced solutions fits the current situation. To this end, we define a *situation descriptor* that allows for comparing tasks and identifying the experience to exploit. Since the ability to generalize to new tasks is largely dependent on how we describe the situation [1], we defined two distinct situation descriptors to deal with global and local situations. A descriptor represents the situation from a navigation point of view and fits the objectives and assumptions considered at each level.

At global level, we define a situation descriptor d_g that consists of the start and goal configurations of the robot expressed in the environment frame. To estimate the similarity between two tasks in this environment, we compute the sum of the Euclidean distances between the start and goal configurations of the robot and consider a threshold to decide at which distance they are similar. This approach is a rather straightforward, but it has been shown to be effective for static environments [2]. This metric allows for multiple-queries on a single environment but provides no information across different environments. This definition of situation descriptor together with the path representation introduced above allows for considering at global level every sub-path contained in a path as a distinct experience. To achieve this, each attractor is stored individually in D_G with a reference to its belonging path. When a new task is requested, the database is searched for the pair of robot configurations (q_i, q_j) such that q_i and q_j belong to the same path P and the sum of the distances to the start and goal configurations

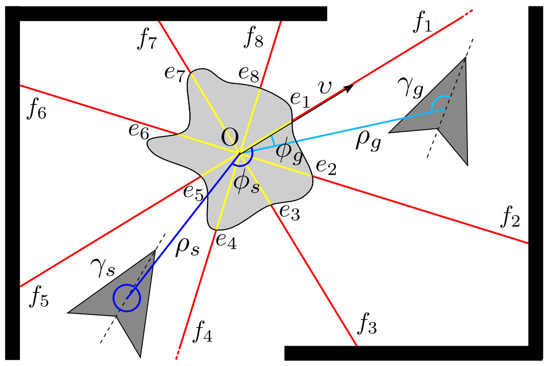


Fig. 2: Description of the local situation. The local descriptor d_l includes the local start (*dark blue*) and goal (*light blue*), the estimate of the structure of the obstacle (*yellow*) and of the extent of free space around it (*red*).

of the new task is the lowest. In this way, the new plan can be guided by any of the sub-paths of P.

At local level instead, we want to generalize local strategies across different environments and obstacles. Hence, the local descriptor d_l needs to be independent from the whole environment and to take into account just local information. We defined it as a vector that consists of 3 main components for a total of 22 attributes, represented in Figure 2.

1) *Local start and goal configurations* are encoded considering the coordinate system (ρ, ϕ, γ) introduced in Section IV that is based only on local information. Referring to Figure 2, $d'_l = \{\rho_s, \phi_s, \gamma_s, \rho_g, \phi_g, \gamma_g\}$.
2) *Shape and dimension of the obstacle* are estimated by computing the distance from the obstacle center O to the obstacle surface in the direction defined by the v vector introduced in the previous section and repeating this measurement in 7 other different directions, each shifted of 45 degrees from each other, $d''_l = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$.
3) *Free space in the local area* is estimated by considering the same directions introduced in the second component and computing for each of them the extents of free space from the obstacle surface to the next obstacle, $d'''_l = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$.

The resulting local descriptor is $d_l = d'_l \cup d''_l \cup d'''_l$. For each good local experience, we compute its local descriptor d_l and store it into the database D_L with a reference to its corresponding attractors. When a new task is requested, D_L is searched for a similar experience comparing the current d_l with the stored ones through the sum of the Euclidean distance between their attributes.

VI. Planning Exploiting the Experiences

Once defined the notions of attractors for representing paths and situation descriptor to compare tasks, we introduce our motion planning system that is able to exploit the good experiences of the robot by incorporating and taking advantage of these concepts. To do so, we employ a modified version of the Attractor Guided Planner (*AGP*) [2] for planning at global and local level.

When a robot is asked for a new task by a user, our system considers: the map of the current environment M, the database D_G of the global paths experienced in M, the database of local paths D_L, the start q_s and a goal q_g configurations of the robot. The main procedure of the system is described in Alg. 1.

Given q_s and q_g, planGlobalLevel computes a global path taking advantage from the previous experiences of the robot in M (Alg. 2). To this end, we compute the global descriptor d_g of the current task as introduced in Section V (line 1). The current descriptor is compared with the descriptors stored in D_G for the given environment M and the attractors $Attr_g$ corresponding to the most similar task are returned (line 2). If a similar task has been experienced, the corresponding attractors are exploited for computing the new path (line 3). To achieve this, we consider a bi-directional RRT that biases its search trees to grow up sampling the attractors $Attr_g$. If one of them is not valid anymore in the current status of the environment, a sample is selected according to a dynamic Gaussian distribution centered in this attractor with a scale growing proportionally to the number of non-valid states sampled around it. In this way, if the robot has experienced a task similar to the current one in a global sense, the algorithm attempts to generate a path that reproduces the structure of the previous solution, even when changes occur in the environment. If no similar task is found instead, a global path is searched from scratch using standard bi-directional RRT that samples uniformly the states from the configuration space (line 5).

Once a global path has been computed, executePath (Alg. 3) enables the robot to execute it, re-planning when needed. In fact, the global level assumes the environment as static, therefore it is possible that some parts of the path are not traversable anymore because of blocking obstacles or changes in the environment. At each step of the robot along this path, we check the remaining path for invalid configurations in the current status of the environment and the first encountered set of those is returned (line 2). When a set of invalid configurations is found, re-planning is triggered to find a local path that enables the robot to avoid the blocking obstacle and to get back to the global path (line 4). This is accomplished by planLocalLevel and the resulting local path is used to update the current path (line 5). When the path is valid, the robot can navigate towards the next path point (line 6).

planLocalLevel is described in Alg. 4 and characterizes the procedure at local level. It starts by estimating the center of mass of the blocking obstacle O (line 1) that allows for computing the descriptor of the current local situation d_l (line 2). The database of local paths D_L is searched for the most similar experienced task and the corresponding attractors are retrieved (line 3). If a similar task is found, the attractors are transformed into the frame of the current environment (line 4) and used to guide the sampling of a new instance of bi-directional RRT (line 5) that works in the same way as in the global level. Otherwise, a new local path is searched using standard bi-directional RRT (line 7).

```
Alg 1 MAIN($q_s, q_g, D_G, D_L, M$)
1: $global\_path \leftarrow$ PLANGLOBALLEVEL($q_s, q_g, D_G, M$)
2: $exec\_path \leftarrow$ EXECUTEPATH($global\_path, q_s, q_g, D_L$)
3: USERFEEDBACK($exec\_path, global\_path, D_G, D_L$)
```

```
Alg 2 PLANGLOBALLEVEL($q_s, q_g, D_G, M$)
1: $d_g \leftarrow$ getGlobalDescriptor($q_s, q_g$)
2: if $Attr_g \leftarrow$ getClosestTask($d_g, D_G, M$) $\neq 0$ then
3:     $global\_path \leftarrow$ biRRT($q_s, q_g, guided\_sampling, Attr_g$)
4: else
5:     $global\_path \leftarrow$ biRRT($q_s, q_g, uniform\_sampling$)
6: return $global\_path$
```

```
Alg 3 EXECUTEPATH($path, q_s, q_g, D_L$)
1: while $p_i \in path \wedge p_i \neq q_g$ do
2:     $invalid\_confs$=$\{p_k, \dots p_{k+j}\}$←getInvalidConfs($p_i, q_g$)
3:     if $invalid\_confs \neq 0$ then
4:         $local\_path$←PLANLOCALLEVEL($p_{k-1}, p_{k+j+1}, D_L$)
5:         $path \leftarrow$ updatePath ($path, local\_path$)
6:     navigateTo($p_i$)
7:     $i \leftarrow i + 1$
8: return $path$
```

```
Alg 4 PLANLOCALLEVEL($q_s, q_g, D_L$)
1: $O \leftarrow$ getObstacleCenter()
2: $d_l \leftarrow$ getLocalDescriptor($q_s, q_g, O$)
3: if $Attr_l \leftarrow$ getClosestTask($d_l, D_L$) $\neq 0$ then
4:     $Attr_l \leftarrow$ toWorldCoord($Attr_l, O$)
5:     $local\_path \leftarrow$ biRRT($q_s, q_g, guided\_sampling, Attr_l$)
6: else
7:     $local\_path \leftarrow$ biRRT($q_s, q_g, uniform\_sampling$)
8: return $local\_path$
```

```
Alg 5 USERFEEDBACK($exec\_path, global\_path, D_G, D_L$)
1: if getFeedback($global\_path$) = good then
2:     $Attr_g \leftarrow$ extractAttractors($global\_path$)
3:     storeExperience($Attr_g, D_G$)
4: $deviations \leftarrow$ getDeviations($exec\_path, global\_path$)
5: for each $dev$ in $deviations$ do
6:     if getFeedback($dev$) = good then
7:         $Attr_l \leftarrow$ extractAttractors($dev$)
8:         $Attr_l \leftarrow$ toLocalCoord($Attr_l$)
9:         storeExperience($Attr_l, D_L$)
```

The presented planning system differs from other approaches since it does not build any graph space [7] or roadmap [5] over the environment. It does not rely on any complex model or maintain hard-coded rules. We adopt a sampling-based approach in which the sampling process is guided by a previous experience. It maintains the flexibility of a general planning system allowing to sample configurations from the whole space when needed. This approach results efficient when planning in (x, y, θ) unlike common planners for robot navigation such as $A*$.

VII. User Feedback

Once the robot has reached the global goal, our system gives the user the possibility to rate its behavior. The user can replay and evaluate through an easy-to-use GUI each robot behavior as *good* or *bad*. As mentioned, only positive experiences are stored and made available to guide planning for new tasks. This enables to implicitly capture the preferences of the user and exploit them when planning in new similar scenarios. USERFEEDBACK (Alg. 5) enables to collect the ratings of the user. First, a feedback is requested for the global path (line 1). If it is regarded as *good*, its attractors are extracted (line 2) and stored in D_G (line 3). Then, *getDeviations* computes the deviations from the global path generated by local re-planning (line 4) and the user can provide a feedback for each of them (line 6). When a local path is rated *good*, its attractors are extracted (line 7), transformed into the local coordinates introduced in Section IV (line 8) and stored in D_L (line 9).

Our planning system does not need for initial data to work. The databases storing the good experiences are built up online while the robot is accomplishing tasks. The higher number of feedback the user will provide, the better the system will fit his preferences.

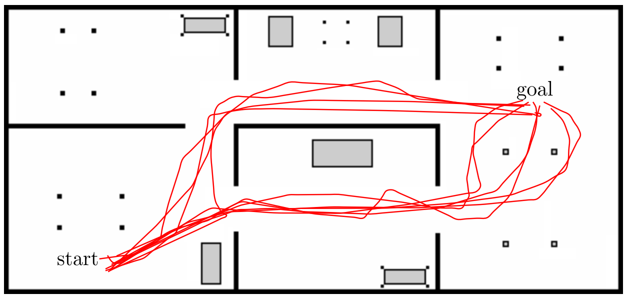
VIII. Experimental Evaluation

The experimental evaluation is designed to illustrate the performance and capabilities of our system as well as to support the main claims made in this paper: providing behaviors for robot navigation that are (i) reproducible over different situations, (ii) predictable for the user, and that (iii) meet his preferences exploiting the previous experiences.

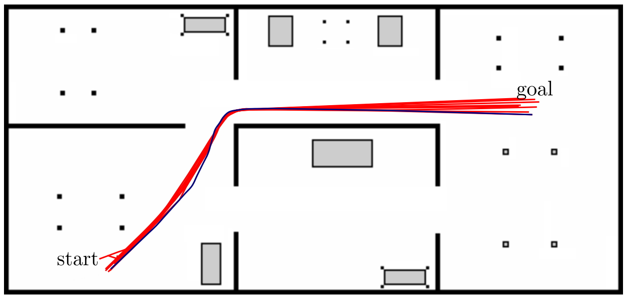
We considered a scenario inspired by the RobDREAM use case to evaluate our work. Here, a human operator requests a simulated KUKA mobile robot to perform several navigation tasks in different environments. We assume the robot has full knowledge of these environments and of the obstacles it will encounter. Throughout this evaluation, we use bi-directional RRT (bi-RRT) as our baseline algorithm.

The first experiment is designed to illustrate how our planning approach works as well as its capabilities. For that, we randomly select in a given environment a set of 10 similar tasks, i.e. planning tasks with similar start and goal configurations, represented in Figure 3. First, we compute a global path for each of these tasks using bi-RRT (Figure 3a). The generated paths reveal large differences with respect to each other: some pass by the top central room and others by the bottom central room. This is due to the sampling-based nature of the algorithm and it does not allow the operator to make any prediction about how a new path for a similar problem will look like or to express any preference about where the robot should move.

To achieve predictable paths and meet user preferences, we use our planning system. In Figure 3b, we consider the same set of similar tasks as considered in Figure 3a and assume the operator prefers the robot passing by the central top room to accomplish them and selects the blue path as *good* example. The paths in red are those generated by our



(a) bi-RRT.



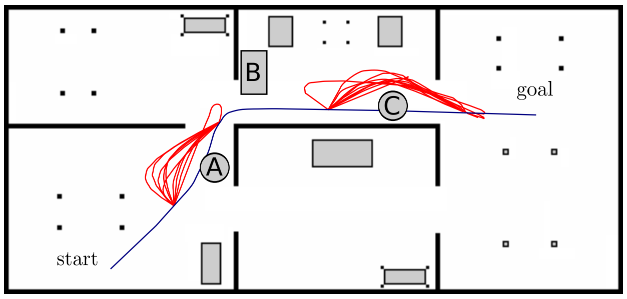
(b) Global level of our system given the blue path as example.

Fig. 3: Global paths generated by standard bi-directional RRT and by our planning system for a set of similar tasks.

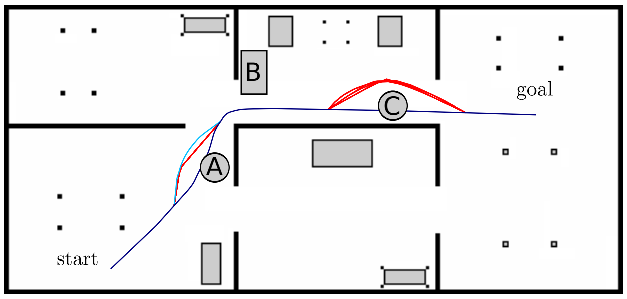
system and reproduce the structure of the *good* path. In this way, it is possible to predict for new similar tasks how the paths will look like and, at the same time, the preference expressed by the operator are satisfied.

To illustrate the local level of our planning system, we consider a robot that is executing the global path provided as example in Figure 3b in a dynamic environment in which unknown obstacles. In Figure 4a, two unforeseen obstacles block the path at two different locations. To allow the robot to reach the goal following the global path, these obstacles have to be avoided and therefore local re-planning is needed. In Figure 4a, the re-planning phase is performed 10 times for each obstacle using bi-RRT. At this level, the difference among the generated paths is smaller than in the global case, however the robot will behave in different ways for distinct planning instances of the same task. To prefer the robot to follow a definite behavior, we consider our planning system in Figure 4b. As in the global case, given the light blue path as example to avoid the obstacle A, the paths generated by our system (red) for obstacle A reproduce the structure of the example. Even though no example is provided for the obstacle C, the paths generated to avoid it reproduce the example provided for obstacle A. This demonstrates that for similar local situations our system is able to reproduce behaviors. Further, the preferences of the operator and the predictability of robot behavior are met at local level as well.

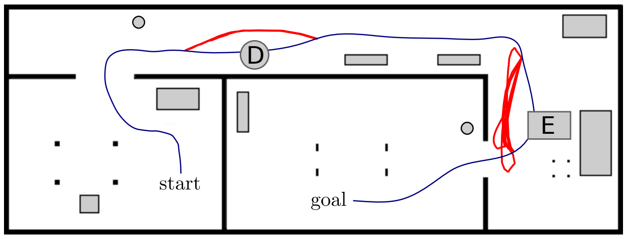
Finally, in Figure 4c, we employ our system considering the same local example provided in Figure 4b in a different environment. Here, the global path (blue) is blocked in two situations. The local situation at obstacle D is similar to the one for the obstacle A, while no similar example is available for the situation at obstacle E. Accordingly,



(a) bi-RRT.



(b) Local level of our system given the light blue path as example.

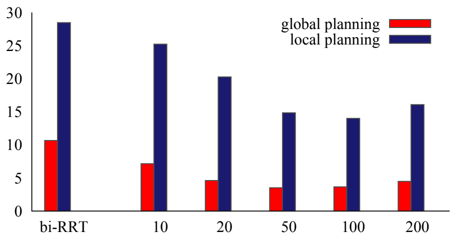


(c) Local level of our system in a different environment given the light blue path in Figure 4b as example.

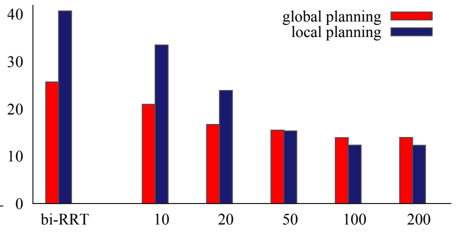
Fig. 4: Local paths generated by standard bi-directional RRT and by our planning system for multiple instances of the same tasks.

the paths generated by our system (red) to avoid the first obstacle reproduce the example, while in the second case the paths are generated from scratch by samplig the states uniformly. This shows the capability of the local level to exploit examples even across different environments and its capacity to generate paths even when no similar example is available with results analogous to bi-RRT.

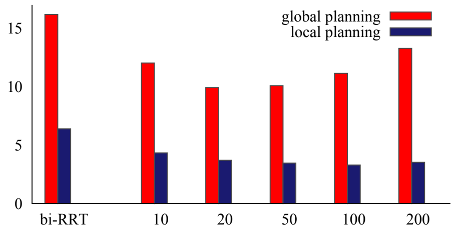
The next set of experiments is designed to illustrate the planning performance of our approach quantitatively. To this end, we considered 20 sets of 10 similar global navigation tasks and 20 sets of 10 similar local tasks in the environments of Figure 3 and 4. At first, we planned for these tasks using bi-RRT. Then, we selected randomly 10, 20, 50, 100 and 200 of these paths both at global and local level as examples for our system and used it to plan for the same tasks. We ran this procedure 10 times for a total of 20,000 planning instances. To evaluate the performance of our system, we considered three quantities: planning time, number of sampled states while planning, percentage of area of the environment covered by the robot when executing a set of similar tasks. We analyze the third quantity to get a measure of how similar are the paths generated for similar situations. The more overlapping is the area covered by the



(a) Planning time (ms).



(b) Number of sampled states.



(c) Area covered by paths for similar tasks (%).

Fig. 5: Performance comparison between bi-directional RRT and our planning system with 10, 20, 50, 100 and 200 examples.

robot to execute two paths, the more similar they are.

Figure 5 shows these quantities both for global and local level for the considered planning settings. Our system outperforms bi-RRT for planning time in every configuration (Figure 5a). The time decreases incrementing the number of examples up to approximately 50% in the local case and 60% in the global case. Since the time for searching the database is proportional to its dimension, when the number of considered examples becomes large, the time tends to increase, but it is still far better than bi-RRT. Figure 5b shows the average number of states that are sampled during planning. When planning for a new task, if a similar experience is available, our system samples the corresponding attractors instead of attempting to explore the whole environment. Therefore, the larger is number of examples, the smaller will be the number of sampled states. Finally, in Figure 5c, we compare the average percentage of area of the environment occupied by the robot when executing each set of similar tasks. Even with few examples, the area covered by our system is approximately 25% smaller than bi-RRT at global level and even 35% at local level. The results at local level are explained by the ability of our approach to generalize experiences across different obstacles and situations. When the number of examples available for each task increases, also the covered area grows correspondingly.

IX. Conclusion

In this paper, we addressed the problem of planning predictable and reproducible paths for mobile robots given user preferences or previous experiences. This is important for navigation in environments where robots share the workspace with the humans. We proposed an easy-to-use motion planning scheme that extends bi-directional RRT and can take into account user preferences during navigation. The preferences are extracted implicitly from the experiences or demonstrations and are automatically considered in the subsequent planning steps through the sampling procedure. We implemented and evaluated our approach on a simulated KUKA mobile robot in different environments. As our experiments suggests, this leads to reproducible and, thus, better to predict navigation behaviors of the robot without requiring experts to hard-coding control strategies or cost functions within a planner. At the same time, our approach typically leads to shorter planning time in cases where previous experiences can be exploited.

References

[1] N. Jetchev and M. Toussaint, “Trajectory prediction: learning to map situations to robot trajectories.” ACM, 2009, pp. 449–456.

[2] X. Jiang and M. Kallmann, “Learning humanoid reaching tasks in dynamic environments,” in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007, pp. 1148–1153.

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

[4] J. J. Kuffner and S. M. LaValle, “Rrt-connect: An efficient approach to single-query path planning,” in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, vol. 2, 2000, pp. 995–1001.

[5] J.-M. Lien and Y. Lu, “Planning motion in environments with similar obstacles.” in *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[6] V. Delsart and T. Fraichard, “Navigating dynamic environments with trajectory deformation,” *CIT. Journal of Computing and Information Technology*, vol. 17, no. 1, pp. 27–36, 2009.

[7] M. Phillips, B. J. Cohen, S. Chitta, and M. Likhachev, “E-graphs: Bootstrapping planning with experience graphs.” in *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[8] D. Berenson, P. Abbeel, and K. Goldberg, “A robot path planning framework that learns from experience,” in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012, pp. 3671–3678.

[9] D. Coleman, I. A. Sucan, M. Moll, K. Okada, and N. Correll, “Experience-based planning with sparse roadmap spanners,” *arXiv preprint arXiv:1410.1950*, 2014.

[10] J. Bruce and M. Veloso, “Real-time randomized path planning for robot navigation,” in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, vol. 3. IEEE, 2002, pp. 2383–2388.

[11] M. Zucker, J. Kuffner, and J. A. Bagnell, “Adaptive workspace biasing for sampling-based planners,” in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*. IEEE, 2008, pp. 3757–3762.

[12] N. Jetchev and M. Toussaint, “Trajectory prediction in cluttered voxel environments,” in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*. IEEE, 2010, pp. 2523–2528.

[13] C. Sprunk, G. D. Tipaldi, A. Cherubini, and W. Burgard, “Lidar-based teach-and-repeat of mobile robot trajectories,” in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 3144–3149.

[14] P. Furgale, P. Krüsi, F. Pomerleau, U. Schwesinger, F. Colas, and R. Siegwart, “There and back again-dealing with highly-dynamic scenes and long-term change during topological/metric route following,” in *ICRA14 Workshop on Modelling, Estimation, Perception, and Control of All Terrain Mobile Robots*, 2014.

[15] D. Perea-Ström, , I. Bogoslavskyi, and C. Stachniss, “Robust exploration and homing for autonomous robots,” *Journ. of Rob. & Aut. Systems*, 2016.

[16] M. Mazuran, C. Sprunk, W. Burgard, and G. D. Tipaldi, “Lextor: Lexicographic teach optimize and repeat based on user preferences,” in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015, pp. 2780–2786.

[17] T. Meriçli, M. Veloso, and H. L. Akın, “A case-based approach to mobile push-manipulation,” *Journal of Intelligent & Robotic Systems*, vol. 80, no. 1, pp. 189–203, 2015.

[18] R. Ros, R. L. De Màntaras, J. L. Arcos, and M. Veloso, “Team playing behavior in robot soccer: A case-based reasoning approach,” in *International Conference on Case-Based Reasoning*. Springer, 2007, pp. 46–60.