The Repetition Roadmap for Repetitive Constrained Motion Planning

P. Lehner; A. Albu-Schäffer

Keywords: Gaussian processes; manipulators; path planning; probability; Repetition Roadmap; repetitive constrained motion planning; Gaussian mixture model; probability distributions; Task analysis; Planning; Manipulators; Service robots; Libraries; Gaussian mixture model; Motion and path planning; mobile manipulation; industrial robots

Citation Notice

```
@ARTICLE{8412538,
author={P. Lehner and A. Albu-Sch\"affer},
journal={IEEE Robotics and Automation Letters},
title={The Repetition Roadmap for Repetitive Constrained Motion Planning},
year={2018},
volume={3},
number={4},
pages={3884-3891},
keywords={Gaussian processes; manipulators; path planning; probability; Repetition Roadmap; repetitive constrained motion planning; Gaussian mixture model; probability distributions; Task
       analysis; Planning; Manipulators; Service robots; Libraries; Gaussian mixture model; Motion and path planning; mobile manipulation; industrial robots},
doi={10.1109/LRA.2018.2856925},
ISSN={2377-3766},
month={Oct},}
```

The Repetition Roadmap for Repetitive Constrained Motion Planning

Peter Lehner Alin Albu-Schäffer

***Abstract*—We present the Repetition Roadmap, a motion planner which effectively exploits the repetitiveness of a set of tasks with small variations to efficiently compute new motions. The method learns an abstract roadmap of probability distributions for the configuration space of a particular task set from previous solution paths. We show how to construct the Repetition Roadmap by learning a Gaussian Mixture Model and connecting the distribution components based on the connectivity information of the prior paths. We present an algorithm which exploits the information in the Repetition Roadmap to guide the search for solutions of similar tasks. We illustrate our method in a maze, which explains the construction of the Repetition Roadmap and how the method can generalize over different environments. We show how to apply the Repetition Roadmap to similar constrained manipulation tasks and present our results including significant speedup in computation time when compared to Uniform and Adaptive Sampling.**

***Index Terms*—Motion and Path Planning, Mobile Manipulation, Industrial Robots**

I. Introduction

Robot manipulators working in flexible manufacturing environments must compute motions to deliver or assemble parts based on perception inputs. Most of these motions must adhere to task constraints, for example not spilling the contents of a box or keeping a drill aligned to an axis. In scenarios where humans and robots collaborate, the robot must adapt its paths to changes induced by the human. But to compete with current, hand-programmed motions, the main constraint of an industrial work-flow is the processing time: Online computation of motions must be fast.

The time constraints of industrial robotics conflict with the fact that planning constrained motions is computationally expensive. The large number of joints in a manipulator form a high-dimensional configuration space. The transformation between the manipulator workspace and the configuration space is non-linear. The task and kinematic constraints form complex composite constraints, which lead to unintuitive manifolds in the configuration space. These difficulties lead to long computation times for motion-planning algorithms.

In industrial manipulation, many of the robot tasks are repetitive. For example, a mobile manipulator fetches boxes from a shelf and delivers them for further processing, as shown in Figure 1. The task may be similar in each case, but slight differences may occur: Changes in the pose of the robot in front of the shelf, in the start configuration of the manipulator, and in the pose of the box already on the robot platform.

Sampling-based planners compute motions by efficiently exploring the configuration space of the manipulator. One

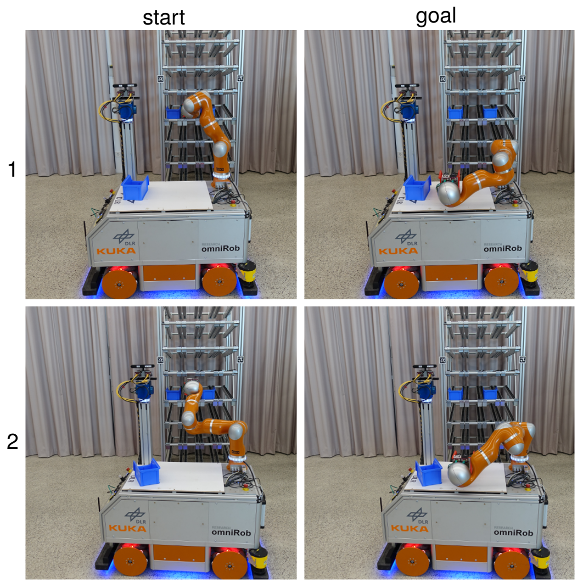

Fig. 1. Two similar tasks of constrained manipulation: Picking a box from an industrial shelf with variations of the platform pose, manipulator start configuration and obstacle box position.

difficulty of the approach is to reuse information from previous similar tasks. To shorten the computation time, multiple new methods inject information from previous solutions to efficiently solve a new planning problem in a similar scenario, including our own contribution, Repetition Sampling [1]. These methods either adapt the sampling strategy of sampling-based planners or construct a discrete roadmap, using previous solutions.

In this paper, we present a novel approach to encode the key information from previous solutions in a *Repetition Roadmap* by combining findings of both adaptive sampling and roadmap approaches. Our method improves on previous algorithms by constructing an abstract roadmap which is a probabilistic representation of the connectivity of the task-relevant configuration space. This does not bind the roadmap to a discrete representation, which has to be evaluated again after each change in the environment, but still provides powerful guidance for the configuration space search. We visualize and show the ability of the algorithm to generalize over significant changes in the environment in a two dimensional maze experiment. Further, we evaluate the method on a constrained manipulation task with our industrial mobile manipulator. The experiments show that Repetition Roadmap's ability to extract and utilize the key information from previous motions to solve new queries significantly faster.

II. Related Work

Sampling-based planning builds mainly on the findings of two core methods. The Probabilistic Roadmap Method (PRM) [2] builds a roadmap in the robots configuration space for multi-query planning. The method is useful for problems where the start and goal configuration change between individual problems, but offers no direct solutions to changes to the topology of the configuration space. The Rapidly-Exploring Random Tree (RRT) [3] for single-query planning implicitly decomposes planning problems with high dimensionality into Voronoi regions. This guides the search tree to unknown regions of the configuration space and afterwards further develops the known regions of the tree. The RRT algorithm has been modified to solve constrained motion-planning problems. The CBi-RRT2 algorithm uses projection and rejection sampling to extend a start and goal tree on the constraint manifold to find such a solution [4].

Our method decomposes the overall search problem into individual local search trees, an approach which has received previous attention. Morales et. al. [5] improve the connectivity of the PRM by growing RRTs between connected components after construction. In a similar fashion Bekris et al. [6] connect the components of the PRM using Bi-RRTs to reduce the number of operations needed during construction. Plaku et. al. [7] parallelize the construction of the PRM with local trees to gain efficiency. Strandberg [8] decomposes a search query into individual RRTs: New trees are spawned at configuration, which cannot be connected to the existing trees. Important for the decomposition into local trees is a) how many trees to grow and b) where to place the root of the individual search trees. Our method provides answers to these questions by exploiting the experience of previous queries and placing the trees at relevant configurations.

Our method adapts the sampling distribution of a sampling-based planner to gain efficiency. Similarly, fixed bias methods include a single Gaussian distribution [9], heuristic sampling to find narrow passages [10], and Voronoi diagram based sampling [11]. Adaptive sampling methods include the Toggle-PRM [12] which constructs one roadmap in the valid configuration-space and one in the invalid configuration-space and adapts the sampling distribution to find narrow passages. Burns et al. [13] introduce the utility-guided approach, learning a distribution to predict the next sample with the highest utility, maximizing the information gain. Informed path sampling [14] adapts the sampling distribution to find paths with high variability. Kobilarov et al. [15] employ the cross-entropy method by learning a Gaussian Mixture Model and optimizing the sampling distribution of one query to converge towards an optimal path. Gammell et. al. [16] iteratively adjust the sampling domain in batches to converge to an optimal solution for a current query. All of these approaches improve over uniform sampling, but do not primarily extract information from previous planning queries.

Planning methods which use information from previously found solutions can be roughly divided into three categories: 1) Path library approaches match complete paths from previous solutions to the new query, 2) roadmap methods build a graph in configuration space to store the connectivity information from previous queries, 3) and adaptive sampling strategies guide the sampling-based planner with information from previous solutions.

1) Path library approaches attempt to match complete paths from previous solutions to the new query. The authors of [17] build a library of trajectories and extract features from the trajectories as input to an imitation learning algorithm. For new queries the algorithm computes the best match to the new features and adapt this solution to the current query. The authors of [18] employ probabilistic inference to select a previous solution path. Other work investigates heuristics to reduce a path set of previous solutions into a path library which is robust with respect to unknown obstacle configurations [19]. Path library approaches can find a solution fast if a matching path can be found, but they do not decompose the prior solution paths and thus cannot construct new solutions from individual elements of multiple paths. This leads to a lower chance of generalizing to new problems, especially problems which are combinations of problems contained in the learning set.

2) Roadmap methods build a graph in configuration space to store the connectivity information already gathered in previous planning queries. The original PRM was optimized to slightly changing environments by the Lazy PRM, which delays collision checks to the actual point of the query [20]. Lazy PRM selects sets of candidate solutions from the PRM and removes candidates to minimize the amount of collision checks. Experience Graphs [21] build a roadmap from the previous solutions and decides based on heuristics if the current search follows the graph or explores a different route. Experience-Based Planning with Sparse Roadmap Spanners aggregates previous solutions in a sparse roadmap [22]. The method evaluates the connections of the sparse roadmap and recomputes colliding paths with a local RRT. Elastic Roadmaps [23] create a roadmap based on visibility around obstacles in the environment. The algorithm adapts the roadmap locally based on changes in the environment to preserve structure. The method was adapted in [24] to iteratively construct the roadmap based on the paths of a sampling-based planner. Prentice and Roy [25] construct a roadmap in belief space which is able to plan paths with expected low uncertainty. Roadmap based methods are efficient at extracting the relevant information of the configuration space, but they represent the connectivity in a discretized roadmap of configurations. For changes in the configuration space, the roadmap must be reevaluated. Additionally, constructing a roadmap for large configuration spaces leads to high memory usage. In contrast, our approach reduces the previous solution paths to an abstract roadmap based on probability distributions, which can compress the key information of previous queries and is not bound to a discrete configuration space representation. We evaluate Thunder [22] in our experiments, to provide a comparison with the presented approach.

3) Adaptive sampling-based approaches guide the sampling-based planner into task-relevant regions of the configuration space. Adaptive workspace biasing learns a sampling distribution using workspace features as input to reinforcement learning [26]. The authors of [27] used the same biasing to simultaneously optimize the planning durations on task and motion layer. Kernel Density Estimation (KDE) based sampling computes a sampling distribution in the configuration space by extracting key configurations from previous solution paths, and integrating kernels based on normal distributions [28]. Ichter et al. [29] use a conditional variational auto encoder to learn sampling distributions for optimal motion-planning. In our previous work, Repetition Sampling [1], we estimated a sampling distribution by extracting key configurations from previous solution paths and learning a Gaussian Mixture Model (GMM) with Expectation Maximization, which was able to capture the correlation between the individual joints. Adaptive sampling strategies can efficiently guide the sampling-based planner to speed up queries, but lack the connectivity information of roadmap based methods. Our method goes beyond adaptive sampling strategies by gathering previous information in an abstract roadmap, thus additionally capturing the connectivity of the configuration space. This allows us to guide *and* decompose the search.

The presented method shares the ideas of learning the relevant configuration space of repetitive tasks, generating a GMM based on key configurations of the previous paths as well as the idea of biasing sampling-based planners with learned distributions with our previous work [1]. Beyond our previous work the RepMap further exploits the connectivity information of previous solution paths by constructing a roadmap of distributions from the GMM: This decomposes the overall problem into local search problems, leading to even faster computation of new queries of a repetitive tasks as shown in the experiments in direct comparison with our previous approach.

III. The Repetition Roadmap

The main motivation of the Repetition Roadmap (RepMap) is to capture the key information of previous solution paths in a roadmap, to guide the search for new queries of a similar task. The RepMap is not a discrete representation, but instead an abstract roadmap of the connectivity of the configuration space based on probability distributions. The RepMap can generalize over different start and goal configurations as well as changes in the environment by remaining on this abstract level. Nevertheless, the information contained in the RepMap is representative enough to provide guidance through the relevant parts of the configuration space.

The RepMap consists of a graph of distributions $G = (V, E)$ in the configuration space C. Each vertex $v_i \in V$ represents a local distribution of task-relevant configurations, approximated by a Gaussian $\mathcal{N}_i$ with the mean μ_i and the covariance Σ_i. Each edge $e_i \in E$ marks a task-relevant connection from one distribution to another.

Algorithm 1: Construct Repetition Roadmap

```
1  Input: A set of $M$ previous solution paths $\mathcal{P}_0 ... \mathcal{P}_M$
2  Output: The Repetition Roadmap $G$
3  foreach path $\mathcal{P}_i$ do
4      Extract key configurations $\mathbf{q}_0 ... \mathbf{q}_n$ from path $\mathcal{P}_i$
5      Insert key configurations $\mathbf{q}_0 ... \mathbf{q}_n$ into learning set $\mathbf{Q}$
6  Compute a GMM based on the learning set $\mathbf{Q}$
7  Extract from the GMM all $K$ Gaussians $\mathcal{N}_0 ... \mathcal{N}_K$
8  foreach Gaussian $\mathcal{N}_k$ do
9      Insert a new vertex $v_k$ into the roadmap $G$
10     Assign the sampling distribution $\mathcal{N}_k$ to $v_k$
11 foreach path $\mathcal{P}_i$ do
12     foreach consecutive configurations $\mathbf{q}_j$ and $\mathbf{q}_{j+1}$ in $\mathcal{P}_i$ do
13         Match $\mathbf{q}_j$ to a Gaussian $\mathcal{N}_k$
14         Match $\mathbf{q}_{j+1}$ to a Gaussian $\mathcal{N}_{k+1}$
15         if $\mathcal{N}_k \neq \mathcal{N}_{k+1}$ then
16             Insert a new edge $e_{new}$ into the roadmap $G$ from
               the respective vertex $v_k$ to $v_{k+1}$
```

When computing a solution based on the RepMap the algorithm initializes a local search tree T_i for each vertex v_i. Once the start and goal configurations of the current query are connected to search trees T_{start} and T_{goal}, the algorithm estimates the most likely path $\mathcal{P}_G$ through G from v_{start} to v_{goal}. Along $\mathcal{P}_G$ the algorithm expands each search tree T_i based on the local distributions $\mathcal{N}_i$ of the respective graph nodes and connects consecutive trees. By focusing the search on establishing the relevant connections in C the RepMap reduces the computation time when finding new solution paths for a repetitive task.

A. Constructing the Repetition Roadmap

We construct the RepMap based on a Gaussian Mixture Model (GMM), a generic model to approximate the density of a distribution. The construction is detailed in Algorithm 1. The individual steps are 1) Extracting key configurations from the previous paths, learning a GMM based on the key configurations (steps 3-7) and 2) Connecting the components of the GMM based on the previous paths (steps 8-16).

1) Learning the Gaussian Mixture Model: We learn the GMM in the same way as described in our previous work, where we used the GMM for adaptive sampling of a Bi-RRT-Connect [1]. From each of the M previous solution path, $\mathcal{P}_j$, we extract key configurations $\mathbf{q}_\mathrm{i}$ and stack these N key configurations in order to construct the learning set $\mathbf{Q}$:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q}_0 & \dots & \mathbf{q}_\mathrm{i} & \dots & \mathbf{q}_\mathrm{N} \end{bmatrix}^T \tag{1}$$

We select the number of components K based on the longest path

$$K = \max_{j=1}^{M} \mathrm{len}(\mathcal{P}_\mathrm{j}), \tag{2}$$

initialize the GMM with k-Means and proceed to learn the GMM based on Expectation Maximation [30]. In each iteration the algorithm calculates the responsibility $r_{\mathrm{i,k}}$ of each Gaussian $\mathcal{N}_{\mathrm{k}}$ describing the configuration $\mathbf{q}_{\mathrm{i}}$ as

$$r_{\mathrm{i,k}} = \frac{w_{\mathrm{k}}\mathcal{N}_{\mathrm{k}}(\mathbf{q}_{\mathrm{i}})}{\sum_{j=1}^{K} w_{\mathrm{j}}\mathcal{N}_{\mathrm{j}}(\mathbf{q}_{\mathrm{i}})} \tag{3}$$

and updates the model mean

$$\mu_{\mathrm{k}} = \frac{1}{R_k}\sum_{i=1}^{N} r_{\mathrm{i,k}}\mathbf{q}_{\mathrm{i}}, \tag{4}$$

variance

$$\Sigma_{\mathrm{k}} = \frac{1}{R_k}\sum_{i=1}^{N} r_{\mathrm{i,k}}(\mathbf{q}_{\mathrm{i}} - \mu_{\mathrm{k}})(\mathbf{q}_{\mathrm{i}} - \mu_{\mathrm{k}})^T, \tag{5}$$

and new weight

$$w_{\mathrm{k}} = \frac{R_k}{N}, \tag{6}$$

of each component where

$$R_k = \sum_{i=1}^{N} r_{\mathrm{i,k}}, \tag{7}$$

until the log likelihood

$$\ln p(\mathbf{Q}) = \sum_{i=1}^{N} \ln\left(\sum_{k=1}^{K} r_{\mathrm{i,k}}\mathcal{N}_k(\mathbf{q}_{\mathrm{i}})\right) \tag{8}$$

reaches a local optimum.

2) Transforming the Gaussian Mixture Model into a Roadmap: This step is one of the main contributions of the presented paper: We construct a roadmap from the GMM which contains probabilistic information about the connectivity of the configuration space for the current task. We initialize the RepMap G with a vertex v_k for each Gaussian $\mathcal{N}_k$ in the GMM and with no edges. To generate the edges we take each path $\mathcal{P}_j$ and match the key configurations to the components of the GMM. When two consecutive configurations $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ on $\mathcal{P}_j$ are matched to two different Gaussians $\mathcal{N}_k$ and $\mathcal{N}_{k+1}$, we connect the corresponding vertices v_k and v_{k+1} in the RepMap with a new edge e_{new}. To determine the effectiveness of the edges we count for each edge how many of the previous solution paths use an edge e_i. We estimate the utility u_{i} that each edge e_i is useful for upcoming planning problems, based on the number of cases e_i was used L_{i} relative to the number of total edges established L:

$$u_{\mathrm{i}} = \frac{L_{\mathrm{i}}}{L} \tag{9}$$

The utility gives an estimation which connections will be useful on future instances of the same task.

B. Planning with the Repetition Roadmap

The RepMap contains information about the relevant configuration space and its connectivity. We use this information on new instances of a task set, to focus the search on the relevant area in the configuration space. The algorithm for using the RepMap in a new planning query consists of the steps detailed in Algorithm 2:

Algorithm 2: Query Repetition Roadmap

```
1  Input: Start configuration $\mathbf{q}_{start}$, goal constraints $S_{goal}$,
   Repetition Roadmap $G$
2  Output: A solution path $\mathcal{P}_{sol}$
3  Initialize each vertex $v_k$ of $G$ with a search tree $T_k$ with the
   sampling distribution $\mathcal{N}_k$
4  Match the start configuration $\mathbf{q}_{start}$ to a Gaussian $\mathcal{N}_{start}$ and
   the respective tree $T_{start}$
5  Grow $T_{start}$ until it connects to $\mathbf{q}_{start}$
6  Find a goal configuration $\mathbf{q}_{goal}$, satisfying the constraints
   $S_{goal}$, with projection sampling.
7  Match the goal configuration $\mathbf{q}_{goal}$ to a Gaussian $\mathcal{N}_{goal}$ and
   the respective tree $Tgoal$
8  Grow $T_{goal}$ until it connects to $\mathbf{q}_{goal}$
9  repeat
10     Compute shortest path $\mathcal{P}_G$: $v_{start}$ to $v_{goal}$ through $G$
11     foreach Edge $e_i$ on $\mathcal{P}_G$ connecting $v_k$ and $v_{k+1}$ do
12         Attempt to connect tree $T_k$ to $T_{k+1}$.
13         if fails then
14             Expland trees $T_k$ and $T_{k+1}$
15             Lower utility $u_i$ of $e_i$: $u_i = \alpha_{\mathrm{u}} * u_i$
16 until $\mathbf{q}_{start}$ is connected to $\mathbf{q}_{goal}$
17 Extract $\mathcal{P}_{sol}$
```

1) Initialize K search trees for the current planning problem at the vertices of the RepMap (step 3).
2) Connect the start configuration to one of the search trees. (steps 4,5)
3) Find a goal configuration which satisfies the goal constraints and connect it to one of the search trees. (steps 6-8)
4) Find the most likely path through the RepMap from the start to the goal configuration. (step 10)
5) Establish connections between the search trees which are the vertices on the most likely path. (steps 9-15)
6) Return the found path which connects the start configuration to the goal configuration through the relevant search trees. (step 17)

1) Initializing the search trees: We initialize a search tree T_k for each vertex v_k of the RepMap. We choose a RRT-Connect search tree with adaptive sampling-based on the associated Gaussian distribution $\mathcal{N}_k$ of v_k. When planning for the constrained manipulation case we select a constrained RRT-Connect, similar to the start tree of the CBi-RRT2 [4]. This allows us to grow T_k limited to the region of interest for this component. We start the tree at the mean μ_k configuration of $\mathcal{N}_k$. If μ_k does not fulfill the constraints of the new task (e.g is in collision), we sample a new configuration from $\mathcal{N}_k$ which fulfills the constraints.

2) Connecting the start configuration: To use the roadmap as a guide to solve the planning problem we first must connect the start configuration $\mathbf{q}_{start}$ to a vertex v_{start} of the RepMap. The algorithm matches $\mathbf{q}_{start}$ to the Gaussian $\mathcal{N}_{start}$ which best represents the start configuration. We then solve the local planning problem by iteratively extending the associated RRT T_{start} of v_{start} until it connects $\mathbf{q}_{start}$, while respecting the task, kinematic, and collision constraints.

3) Finding and connecting the goal configuration: Next we find and connect the goal configuration $\mathbf{q}_{goal}$ to the RepMap. The algorithm uses projection sampling to find $\mathbf{q}_{goal}$ and matches it to the Gaussian $\mathcal{N}_{goal}$ which best represents $\mathbf{q}_{goal}$. We then solve the local planning problem by iteratively extending the associated RRT T_{goal} of v_{goal} until it connects to $\mathbf{q}_{goal}$, while respecting the task, kinematic, and collision constraints.

4) Finding the most likely path as guidance: Once the algorithm has found a connection to the RepMap from the start and goal configurations it searches for the most likely path through the RepMap. To estimate the most likely path the algorithm computes the cost for each edge C_i as

$$C_\mathrm{i} = \log\left(\frac{1}{u_\mathrm{i}}\right), \tag{10}$$

which represent a zero connection cost when u_i is one and goes towards an infinite cost when the u_i is close to zero. Based on the edge costs C_i we compute the shortest path from the start to the goal configuration using the standard Dijkstra algorithm. The found path $\mathcal{P}_G$ represents the path through the roadmap which is estimated to have the highest probability of connecting the start and goal configurations.

5) Connect the search trees on the path: To solve the current planning problem we need to translate the abstract path $\mathcal{P}_G$ through the RepMap into a a sequence of configurations which respect the task, kinematic, and collision constraints. The algorithm iteratively selects one unconnected edge e_i represented by vertices v_k, v_{k+1} from the $\mathcal{P}_G$. The algorithm tries to connect the associated trees T_k and T_{k+1}. If the connection fails we extend both trees and update the $\mathcal{P}_G$ as described in the last paragraph.

6) Extracting the solution path: Once all the edges on the abstract path $\mathcal{P}_G$ through the RepMap have been established, a path of configurations exists from the start to the goal configuration. We extract this solution path $\mathcal{P}_{sol}$ and return it as the solution to the current planning problem.

7) Updating the most likely path: The most likely path $\mathcal{P}_G$ through the RepMap does not always provide the best guidance for the planning query, due to the variations. To reflect the information already gathered for the current query, we update the utility of the edges. If a connection attempt between two RRTs fails, we scale the utility u_i with a factor α_u:

$$u_\mathrm{i}{}^{j+1} = \alpha_\mathrm{u} u_\mathrm{i}{}^{j}, \qquad (0 < \alpha_\mathrm{u} < 1) \tag{11}$$

After each update we recompute the most likely path $\mathcal{P}_G$. α_u balances the exploitation and exploration of the RepMap and indicates how much the algorithm trusts the prior connectivity information in the RepMap. In our experiments we choose $\alpha_\mathrm{u} = 0.8$.

IV. Experiments

We illustrate the properties of the RepMap with two experiments: The two dimensional maze example illustrates the creation and usage of the RepMap and illustrates the ability to generalize over large variations, while the constrained manipulation experiment shows the effectiveness of the method in a realistic industrial use case. In both experiments we compared the RepMap with a Bi-RRT with Uniform Sampling, a Bi-RRT with Repetition Sampling and to the Thunder framework [22], a sampling-based planner which constructs a discrete roadmap from previous solution paths. We adapted the Open Source implementation which was available through the Open Motion Planning Library and integrated our collision checker to achieve comparability. All data were recorded on an Intel® Xeon® 3.60 GHz CPU with 8 GB RAM. To ensure the statistical relevance of the data we computed a pairwise t-test between the computation times of all planners. The largest p-value of all t-tests is 0.02 which indicates the data is statistical relevant.

A. Maze

We present a maze experiment to visualize the RepMap in a two dimensional example and to show that the method can generalize information from previous solutions to similar but significantly different tasks.

We designed a basic maze as shown in Fig. 2 a) and performed multiple evaluations where we added random obstacles (circles) to the maze ranging from 10 to 120 additional obstacles. In each evaluation we calculate 200 solution paths on each of the learning scenarios (b). From the solution paths we extract the key configurations and learn a GMM (c). The algorithm classifies the key configurations to the corresponding components of the GMM and connects them to form the RepMap (d). To validate the RepMap we compute solutions for 100 different scenarios e).

The overall computation time results are depicted in Fig. 5. In all cases the RepMap was at least twice as fast as Uniform Sampling. In most cases the RepMap was only slightly faster than Repetition Sampling but had significant lower variance in cases with many random obstacles. The Thunder Framework performed faster in cases with few random obstacles, but slower in cases with many random obstacles.

Figure 3 shows a more detailed analysis for the cases of ten and one hundred random obstacles. With ten random obstacles Thunder was roughly ten times faster than the RepMap in the median case. In the case of one hundred random obstacles the RepMap was about four times faster than Thunder in the median case and the standard deviation of Thunder was eight times larger.

To gain insights about path quality, Figure 4 compares the relative path lengths for the case of one hundred obstacles. The comparison methods had a path length of about 1.5 to 2 times the width of the maze in the average case while the RepMap had a path length of about 2.75 the width of the maze in the average case.

B. Constrained Manipulation Tasks

To evaluate the RepMap on a real robotic platform we choose a task from industrial mobile manipulation. The robots task is to pick a box from an industrial shelf and place it on the

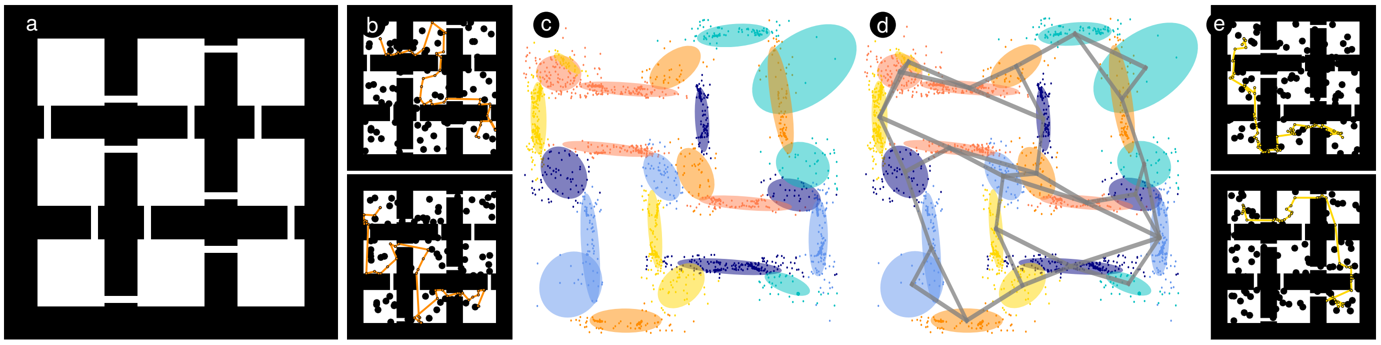

Fig. 2. The maze experiment in two dimensions (start upper left, goal lower right) for the case of one hundred random obstacles. During each evaluation we place random obstacles into the basic maze (a, b) and compute paths on each of the learning scenarios b) (two shown). Based on the learning paths we compute a GMM c) and construct the Repetition Roadmap d). Based on the roadmap the algorithm computed solution paths for one hundred validation examples e) (two shown).

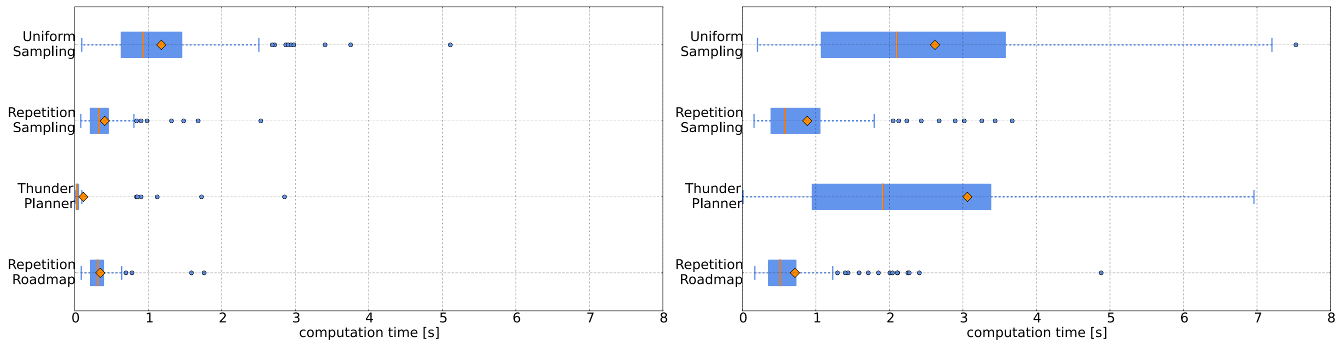

Fig. 3. Validation set computation time of the simple maze experiment (mean marked with diamond) with ten (left) and one hundred (right) obstacles.

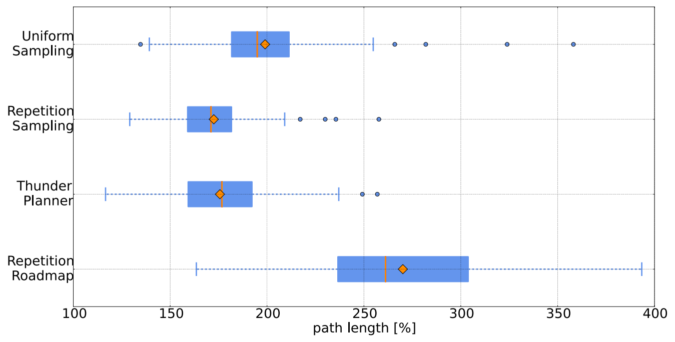

Fig. 4. Validation set path length of the maze experiment (mean marked with diamond) for the case of one hundred obstacles. Path length given in percent of maze width.

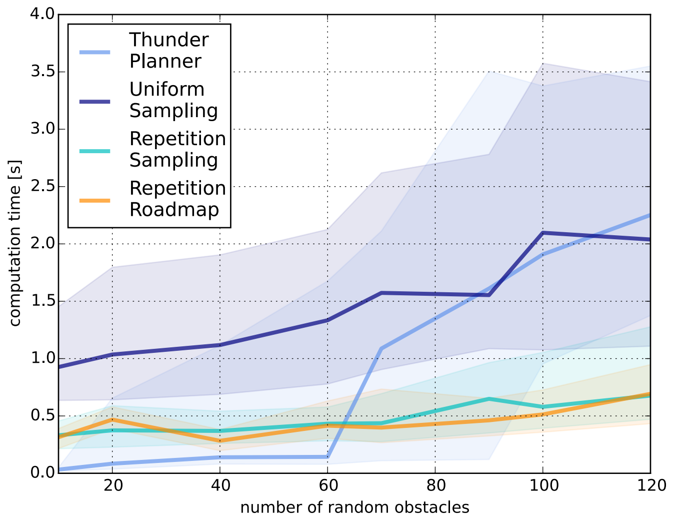

Fig. 5. Plot of the validation set computation time of the maze. The solid line shows the median computation time and the shaded regions show the lower and upper quantiles.

platform of the robot, as shown in Figure 6. The box contains parts for assembly at a second station and has to be kept level to not spill the contents. For each instance of the task we included three variations to the setting: 1) The pose of the mobile platform differs each time in front of the shelf, 2) the pose of a second box on the robot platform changes, and 3) the start configuration of the manipulator is different in each instance. We plan the manipulator motion decoupled from the platform motion, since synchronization of the movements is not possible in the current hardware setup.

We generated a learning set of one thousand solutions paths by solving instances with a CBi-RRT2 with Uniform Sampling. Based on these paths we fitted a GMM and created the RepMap as shown in Figure 6 b). We evaluated all planners on a set of 300 validation tasks. The comparison results are shown in Figure 7. The RepMap is about three times faster than Uniform Sampling, about two times faster than Thunder and about 1.3 times faster than Repetition Sampling in the average case. Additionally the variance in computation time of the RepMap is at least 1.5 times smaller than the variances of the compared approaches.

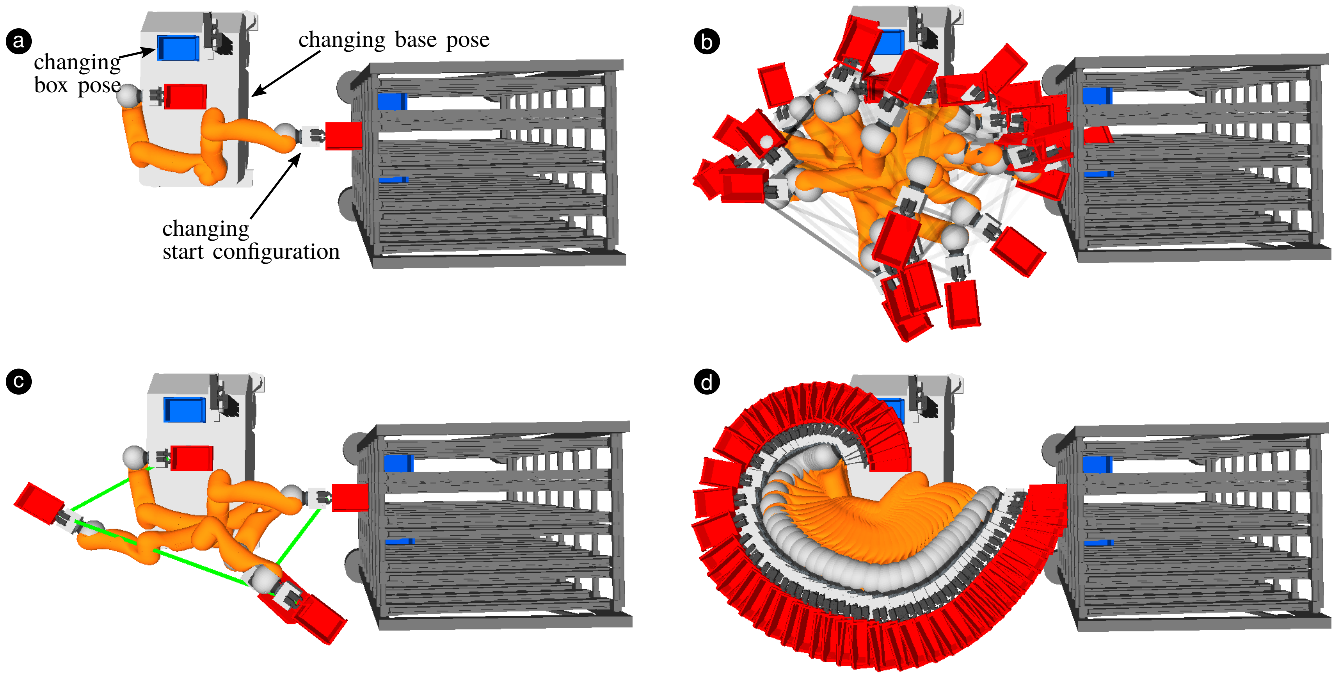

Fig. 6. Illustration of the constrained manipulation experiment, shown as a top view of Figure 1. The task of the mobile manipulator is to pick a box from a shelf, while not spilling the contents. We vary the tasks in the pose of the robots base, the pose of a box on the platform of the robot and the start configuration of the arm as shown in a). Based on the learning paths we compute the Repetition Roadmap b) and used it to guide the configuration space search along the most likely path c) (the configurations represent the means of the distributions and the lines indicate the connections of the roadmap and the path). The respective smoothed solution path of configurations for this instance of the task is shown in d)

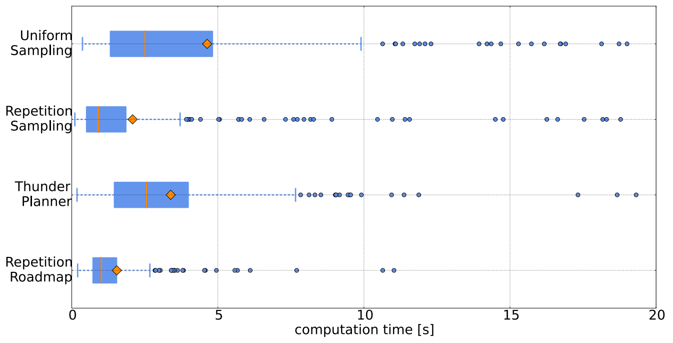

Fig. 7. Boxplot of the validation set computation time of the constrained manipulation experiment (mean marked with diamond).

V. Discussion

The experiments show that the RepMap can a) generalize over significant changes in the planning problem and b) leverage information from previous solutions to solve new problems of a similar kind significantly faster.

In both experiments the RepMap was able to generalize over significant changes. In the maze experiment the method was able to generalize over many random obstacles. In the constrained manipulation experiment the RepMap was able to generalize over three changes: The changing base position of the robot, the changing start configuration of the manipulator and the changing obstacle box position on the robot platform.

In both experiments the RepMap was able to compute solutions for the validation tasks faster than a RRT with Uniform Sampling or Repetition Sampling. In the low dimensional configuration space of the maze experiment and especially in the constrained manipulation experiment the RepMap was faster than Uniform or Repetition Sampling and had a lower variability in computation time. A reduction of the computation time of the collision checks by a factor of five to ten seems feasible with newest software tools based on GPU parallelization, bringing the method to a computation time range which is desirable in industrial environments.

The RepMap was slower than Thunder in cases with low variability but faster in cases with high variability. Thunder discretizes the configuration space into a roadmap. Depending on the placement of the random obstacles the roadmap does not contain useful connectivity information. In these cases Thunder uses an Uniform Bi-RRT planner to generate a solution, which was the case in five percent of the validations with ten obstacles, but in ninety percent of the validations with on hundred obstacles. In cases with high variability the computation time of Thunder approaches the computation time of Uniform Sampling. The variability of the manipulation experiment was seemingly high as about eighty percent of queries were solved by the Uniform-RRT and only twenty percent were solved with the help of the sparse roadmap. The RepMap and Repetition Sampling were able to encode and exploit the connectivity information for cases with high variability by learning a *probabilistic distribution* of the relevant configuration space.

The results of the maze experiment indicate that the solutions found by the RepMap have a lower path quality than of the comparative methods. We believe this is due to the fact that in its current instantiation the cost of the graph search is purely based on expected connectivity. The guiding path of search trees biases the discrete search towards robust areas of the configuration space, irrelevant of the path cost. To balance the likeliness of finding a valid path and path cost we suggest to formulate a hybrid cost function which considers both aspects.

The underlying assumption of the RepMap is that the new query is similar to the queries of the learning paths. This limits the usability of the RepMap in cases where the environment and the task change drastically. Major changes to the environment, for example rotating the whole maze by ninety degrees or placing a large obstacle in between the shelf and the robot, will slow the guidance of the RepMap. Major changes to the task, for example introducing a conflicting task constraint with a new grasp, will change which parts of the configuration space are relevant and diminish the RepMap's ability to guide the search. When fundamentally redesigning a task, for example by significantly changing the end-effector constraints or changing the type of environment, a new RepMap should be generated from a corresponding learning set.

The RepMap is designed to focus the search on the relevant parts of the configuration space for a particular task. It balances exploration of the configurations space with exploitation by updating the most likely path with the information from the current query. Nevertheless, by guiding the RRTs with the contained distributions it will not find a path to any general planning problem, and thus has no probabilistic completeness guarantees. If probabilistic completeness is a desired property for the application, we propose two adaptations of the underlying algorithm. The first adaptation uses a Uniform RRT in parallel to the RepMap algorithm as suggested by [22]. This ensures the probabilistic completeness guarantees of the Uniform RRT and the new solutions gained from the Uniform RRT can enrich the experience of the RepMap. The second adaptation iteratively broadens the Gaussian distributions of the RepMap and the underlying sampling distributions of the search trees converge towards uniform distributions, establishing the prerequisite for probabilistic completeness.

VI. Conclusion

In this paper we presented the Repetition Roadmap, a motion-planning algorithm to encode previous planning experience in an abstract roadmap, which we query to efficiently partition the search into local planning problems and guide the search through the task-relevant parts of the configuration space. We evaluated the approach in a maze example to visualize the method and to show how the roadmap can generalize over significant task variations. Furthermore, we evaluated the algorithm on a real constrained motion-planning problem with an industrial mobile manipulator and were able to show that the computation times of the RepMap were significantly shorter than those of a standard RRT, those of our previous contribution, Repetition Sampling and those of the Thunder Framework. In future work we would like to investigate an autonomous clustering of similar tasks, to enable the robot to decide which tasks are similar.

References

[1] P. Lehner and A. Albu-Schäffer, "Repetition sampling for efficiently planning similar constrained manipulation tasks," in *Proc. 2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 2851–2856.

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

[3] S. M. LaValle, "Rapidly exploring dense trees," *Planning Algorithms*, pp. 228–237, 2004.

[4] D. Berenson, S. S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[5] M. Morales, S. Rodriguez, and N. M. Amato, "Improving the connectivity of PRM roadmaps," in *Proc. 2003 IEEE Int. Conf. Robotics and Automation (ICRA)*, vol. 3, Sept 2003, pp. 4427–4432.

[6] K. E. Bekris, B. Y. Chen, A. M. Ladd, E. Plaku, and L. E. Kavraki, "Multiple query probabilistic roadmap planning using single query planning primitives," in *Proc. 2003 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, vol. 1, Oct 2003, pp. 656–661.

[7] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki, "Sampling-based roadmap of trees for parallel motion planning," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 597–608, Aug 2005.

[8] M. Strandberg, "Augmenting RRT-planners with local trees," in *Proc. 2004 IEEE Int. Conf. Robotics and Automation (ICRA)*, vol. 4, April 2004, pp. 3258–3262.

[9] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *Proc. 1999 IEEE Int. Conf. Robotics and Automation*, vol. 2. IEEE, 1999, pp. 1018–1023.

[10] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *Proc. 2003 IEEE Int. Conf. Robotics and Automation*, vol. 3. IEEE, 2003, pp. 4420–4426.

[11] Y. Yang and O. Brock, "Adapting the sampling distribution in PRM planners based on an approximated medial axis," in *Proc. 2004 IEEE Int. Conf. Robotics and Automation*, vol. 5. IEEE, 2004, pp. 4405–4410.

[12] J. Denny and N. M. Amato, "Toggle PRM: Simultaneous mapping of C-free and C-obstacle - a study in 2D," in *Proc. 2011 IEEE Int. Conf. Intelligent Robots and Systems*. IEEE, 2011, pp. 2632–2639.

[13] B. Burns and O. Brock, "Toward optimal configuration space sampling." in *Robotics: Science and Systems*, 2005, pp. 105–112.

[14] R. A. Knepper and M. T. Mason, "Real-time informed path sampling for motion planning search," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1231–1250, 2012.

[15] M. Kobilarov, "Cross-entropy motion planning," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 855–871, 2012.

[16] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proc. 2015 IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2015, pp. 3067–3074.

[17] 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.

[18] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *Proc. 2012 IEEE Int. Conf. Robotics and Automation*. IEEE, 2012, pp. 3671–3678.

[19] M. S. Branicky, R. A. Knepper, and J. J. Kuffner, "Path and trajectory diversity: Theory and algorithms," in *Proc. 2008 IEEE Int. Conf. Robotics and Automation*, May 2008, pp. 1359–1364.

[20] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *Proc. 2000 IEEE Int. Conf. Robotics and Automation*, vol. 1. IEEE, 2000, pp. 521–528.

[21] M. Phillips, B. J. Cohen, S. Chitta, and M. Likhachev, "E-graphs: Bootstrapping planning with experience graphs." in *Robotics: Science and Systems*, vol. 5, no. 1, 2012.

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

[23] Y. Yang and O. Brock, “Elastic roadmaps—motion generation for autonomous mobile manipulation,” *Autonomous Robots*, vol. 28, no. 1, pp. 113–130, 2010.

[24] P. Lehner, A. Sieverling, and O. Brock, “Incremental, sensor-based motion generation for mobile manipulators in unknown, dynamic environments,” in *Proc. 2015 IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2015, pp. 4761–4767.

[25] S. Prentice and N. Roy, “The belief roadmap: Efficient planning in belief space by factoring the covariance,” *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1448–1465, 2009.

[26] M. Zucker, J. Kuffner, and J. A. Bagnell, “Adaptive workspace biasing for sampling-based planners,” in *Proc. 2008 IEEE Int. Conf. Robotics and Automation*. IEEE, 2008, pp. 3757–3762.

[27] R. Chitnis, D. Hadfield-Menell, A. Gupta, S. Srivastava, E. Groshev, C. Lin, and P. Abbeel, “Guided search for task and motion plans using learned heuristics,” in *Proc. 2016 IEEE Int. Conf. Robotics and Automation*. IEEE, 2016, pp. 447–454.

[28] T. F. Iversen and L.-P. Ellekilde, “Kernel density estimation based self-learning sampling strategy for motion planning of repetitive tasks,” in *Proc. 2016 IEEE Int. Conf. Intelligent Robots and Systems*. IEEE, 2016, pp. 1380–1387.

[29] M. P. B. Ichter, J. Harrison, “Learning sampling distributions for robot motion planning,” in *Proc. 2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2018, pp. 7087–7094.

[30] C. Bishop, “Mixture models and EM,” *Pattern Recognition and Machine Learning*, pp. 423–455, 2006.