Balancing Exploration and Exploitation in Sampling-Based Motion Planning

Markus Rickert, Arne Sieverling, and Oliver Brock

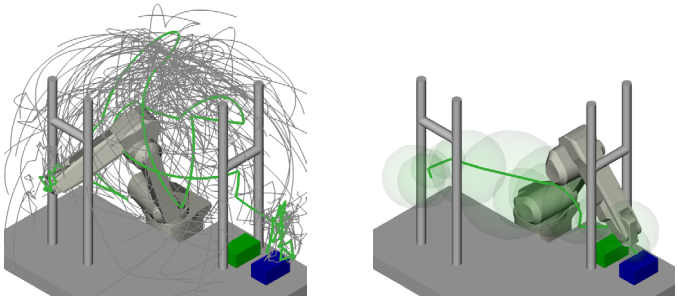
***Abstract*—We present the Exploring/Exploiting Tree (EET) algorithm for motion planning. The EET planner deliberately trades probabilistic completeness for computational efficiency. This trade-off enables the EET planner to outperform state-of-the-art sampling-based planners by up to three orders of magnitude. We show that these considerable speed-ups apply for a variety of challenging, real-world motion planning problems. The performance improvements are achieved by leveraging work space information to continuously adjust the sampling behavior of the planner. When the available information captures the planning problem's inherent structure, the planner's sampler becomes increasingly exploitative. When the available information is less accurate, the planner automatically compensates by increasing local configuration space exploration. We show that active balancing of exploration and exploitation based on workspace information can be a key ingredient to enabling highly efficient motion planning in practical scenarios.**

***Index Terms*—Path Planning for Manipulators, Sampling Strategy, Balancing Exploration and Exploitation**

I. Introduction

SAMPLING-BASED approaches are the de facto standard in motion planning today. Their continued development, paired with increased available computational power, has led to the application of planning methods to new and complex problem domains. Throughout much of this exciting development, the notion of probabilistic completeness was an important concern for the design of new algorithms. Probabilistic completeness attests to an algorithm's ability to solve "any" planning problem, provided a solution exists and sufficient computational resources are available.

This paper is based on the view that probabilistic completeness is not a strong indication of the real-world effectiveness of a motion planner. For one, it is often easy to turn a sampling-based motion planning algorithm, even a poorly conceived one, into a probabilistically complete algorithm: It suffices to add a small percentage of random samples to the sampling of the planner to obtain this characteristic. Thus, probabilistic completeness is not a good indication of a planner's quality. Second, it has been proven that the ability to solve any solvable planning problem (as required for probabilistic completeness) may lead to computational costs exponential in the dimensionality of the configuration space [1]. This is a consequence of the necessity to solve *all* possible planning problems, even the hardest ones, and even if they do not occur in the real world. This prevents the planner from making assumptions about the planning problem that are generally true in the real world and can lead to increased computational efficiency. Third, if a planner is applied to practical planning problems, the value of completeness guarantees seems to vanish: They only remain valid for the exact world model used for planning and under the assumption of perfect actuation. In dynamic and unstructured environments, neither of these are available. We aim to demonstrate that, from a practical perspective, the notion of completeness should not be a requirement for a motion planner.



(a) (b)

Figure 1. Illustrating the beneficial effect of balancing exploration and exploitation in motion planning: End-effector trajectories (*gray*) indicate the amount of configuration space explored; the solution path is shown as a *green* line; the workspace information used for exploitation is shown as *green* transparent spheres. (a) A single-query planner with guided exploration (ADD-RRT) explores large amounts of configuration space, (b) the exploring/exploiting tree (EET) algorithm described in this paper exploits workspace information to guide sampling and as a result has to perform very little exploration.

In this paper, we present a motion planner for real-world planning problems, i.e., planning problems that contain considerable exploitable structure (Fig. 1). The planner is deliberately incomplete: It might fail even when a solution exists! But it outperforms existing sampling-based planners by up to three orders of magnitude in a variety of realistic and challenging planning problems. It even fails less often than the existing sampling-based planners we compared it to, if failure is defined as not solving a problem within a reasonable amount of time.

Our planner is based on a very simple insight: If motion planners should avoid searching the entire configuration space (which would incur an exponential computational cost), they must assess which regions of configuration space are relevant to the problem and which are not. By excluding parts of the configuration space, planners can become computationally

M. Rickert is with fortiss GmbH, An-Institut Technische Universität München, Munich, Germany.

A. Sieverling and O. Brock are with the Robotics and Biology Laboratory, Faculty of Electrical Engineering and Computer Science, Technische Universität Berlin, Germany. They were supported by the Alexander von Humboldt foundation, the Federal Ministry of Education and Research (BMBF), and the European Commission (FP7-ICT-248258-First-MM).

more efficient—and incomplete, by design!

To select relevant configuration space regions, the planner must use information. If this information is highly accurate, the planner should exploit it to direct search in the configuration space. When the information is inaccurate, the planner's success will depend on its ability to realize this and to complement the exploitation of information with explorative search.

The planner presented in this paper acquires information from the workspace description of the planning problem and uses this information to direct the search in configuration space. The search balances exploration and exploitation, depending on the quality of the acquired information. We evaluate this planner based on comparative performance analysis with other state-of-the-art motion planners.

The technical content of this paper is based on prior publications [2], [3]. Here, we present a refined formulation of the EET algorithm. The changes are described in Section III-D. We also present a completely new experimental evaluation, demonstrating the strength of the planner on a broad range of problems from industrial robotics, mobile manipulation, and computational biology. The source code of the EET algorithm, the planners used for comparison, and the experimental scenarios are now publicly available.

II. Related Work

The proposed planner relies on the use of information to guide configuration space exploration. The discussion of related work will therefore examine how existing planners use information for this purpose. We can distinguish different levels of information usage.

- **Exploitation:** The planner determines a part of the plan based on available information with the sole objective of advancing the plan toward the solution. An example of this behavior are simple artificial potential field methods.
- **Exploration:** The planner uses no information to sample the configuration space. The goal of sampling is to gain understanding of the space, not to directly find a solution. An example of this behavior can be found in the uniform random sampling of PRM planners and in the Voronoi-bias of RRT planners (expand toward unexplored regions rather than toward a solution).
- **Guided exploration:** The planner uses information to select configuration space regions in which to search for a plan. These regions are selected based on information about possible solution paths. Medial-axis PRM planners fall into this category.

We will now identify these three levels of information usage in related approaches.

A. Gradient Descent

Gradient descent methods exploit the information represented in a potential function to generate robot motion. The effectiveness of gradient descent depends on the information captured by the potential function. Artificial potential fields [4] rely on local proximity information to obstacles and distance to the goal location. This information is easy to derive from sensor data or from a geometric world model. Hence, artificial potential field methods are computationally efficient and suitable for reactive motion. However, they are also susceptible to local minima and saddle points in the potential.

Navigation functions [5] are local minima-free potential functions. They avoid local minima by considering global information. In realistic settings, the computation of this global information requires a complete exploration of the configuration space [6]. Once this complete exploration has been performed, motions toward a fixed goal configuration in a static environment can be determined with pure exploitation. Complete navigation functions become impractical in high-dimensional configuration spaces or in dynamic environments. However, they might be the earliest approaches that combine exploration and exploitation.

B. Sampling-Based Motion Planning

The Probabilistic Roadmap (PRM) planner [7], [8] creates an initial roadmap using pure exploration: Each sample is placed uniformly at random, i.e., completely uninformed. In a subsequent refinement phase, the planner performs guided exploration to connect the different components of the resulting roadmap. Hence, even the earliest examples of sampling-based motion planners combined some forms of exploration and exploitation, albeit in a fixed and sequential manner.

Rapidly-exploring random tree (RRT) planners [9] solve a particular planning task by incrementally growing a tree, starting from a specific location. Configuration space exploration is guided toward the largest Voronoi region associated with the existing samples. This drives exploration toward large unexplored regions. As the Voronoi regions of samples become approximately equal in size, the exploratory behavior gradually shifts from expansion of the tree to refinement.

The Voronoi bias, while yielding desirable theoretical properties for the RRT, attempts to explore the entire configuration space without any goal-directed bias. It is therefore very critical to combine the exploratory behavior of the Voronoi bias with the exploitative connect step, introduced as an extension of the algorithm very early on [10]. The connect step simply attempts to connect the newest sample directly to the goal location. This alternation of exploration and exploitation lets RRT planners solve many high-dimensional motion planning problems time-efficiently.

Researchers soon recognized the benefits of informed sampling strategies for PRM planners. To give some examples: Gaussian sampling [11] and obstacle-based PRM [12] place samples close to obstacle boundaries. The bridge test [13] increases sampling density in narrow passages and visibility-based PRM [14] controls the placement in regions already covered by existing samples. Other planners combined several sampling strategies. These planners select the most suitable heuristic for a region of configuration space [15], [16]. They effectively leverage information obtained about those configuration space regions to adapt the sampling strategy.

All of these sampling strategies—and many others similar to these—exploit information obtained during sampling and use fixed heuristics derived from intuitions about the structure of configuration space. As experimental evaluations in the lit-

erature show, these fixed, local heuristics can improve planner performance but also lead to pathologies.

The next generation of sampling methods were inherently adaptive, thereby overcoming the limitations of using a finite set of fixed samplers. These adaptive methods also included the information contained in colliding samples; past approaches had simply discarded this information. These sampling methods view the collection of acquired free and colliding samples as a non-parametric model of configuration space. This model can support powerful methods of selecting next samples, given the available information [17]–[19].

Similar adaptive methods were also applied to RRT based planners. The Voronoi bias of RRT planners leads into a type of pathology called the "bug trap problem". It describes the difficulty of Voronoi-based exploration to escape enclosed spaces with only narrow openings. A number of modified expansion heuristics have been developed so as to alleviate this effect [20]–[24]. These heuristics stop the extension of samples for which extension has failed repeatedly, effectively exploiting additional information about the local configuration space of the sample.

The progressive development of sampling methods supports the perspective taken in this paper: Information must be leveraged during planning. The more information we have, the more efficient our planners can be. Note that the improvements are the result of an active selection of configurations space regions to explore, i.e., the result of guided exploration.

C. Planners Using Workspace Information

To devise effective planners, it is not only important to consider *how* information should be used, we must also consider *what* information to use. The most obvious information to use for guiding configuration space exploration or for performing exploitation is workspace information. This information is readily available in motion planning or can easily be acquired through sensors. We will discuss how planners presented in the literature have used this information source.

The most common use of workspace information is to guide exploration. Planners vary in their method of extracting workspace information. Methods include medial axes or generalized Voronoi diagram [25]–[28], Watershed segmentation [29], Delaunay triangulation combined with an adaptive hybrid sampling approach [30], approximation of the medial axis by using partially overlapping maximum spheres [31], or tunnels of free workspace, also computed by a sphere expansion method [32]. In a similar approach, the workspace is divided into discrete regions and connectivity information for those regions is used to guide planning [33]. RRT planners were adapted to explore the task-space uniformly [34], which is advantageous in very high-dimensional configuration spaces. Disassembly-based motion planning [35] identifies narrow passages based on the available free workspace. A configuration inside the narrow passage is found using uniform sampling of a small configuration space region. Given such a configuration, diffusion-based disassembly easily finds solution to sub-problems, the concatenation of which yields a solution to the overall problem.

This discussion shows that workspace information is a rich and versatile source of information for efficient motion planning.

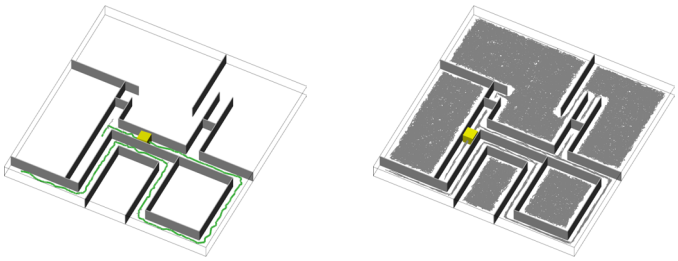
III. Exploring/Exploiting Trees

This section describes the Exploring/Exploiting Tree (EET) planner, a tree-based motion planning algorithm that performs exploitation whenever possible and gradually transitions to exploration when necessary. The planner is based on a single tree expansion in configuration space, similar to RRT methods [9], [10]. The expansion is guided in the task space. This enables the planner to solve a task frame specification in workspace rather than a specific goal configuration [36], [37]. The EET's main design objective is to carefully balance exploratory and exploitative behavior so as to leverage the structure inherent in the planning problem for rendering motion planning as efficient as possible. In order to accomplish this, we designed the EET to behave like a potential field planner whenever possible and to gradually turn into a nearly-complete motion planner when required. We will present a mechanism for gradually shifting between exploration and exploitation based on workspace information and a way of gathering this information. We now will introduce the technical contribution of the EET at a high level and then give an algorithmic description in the second part of this section. Table I provides a summary of symbols and variables used in this section.

A. Algorithm Overview

Consider a maze with long narrow corridors similar to Fig. 2. A free-flying robot in the shape of a box should ideally exploit the structure of its surroundings and follow the corridors without much rotation until it reaches a turn. However, due to its shape, moving through the tight corners is difficult and can only be achieved through exploration. Sampling-based approaches in configuration space will be able to create a path through the turns, their explorative behavior will, however, also waste considerable time exploring paths that bump into the walls of the maze.

The EET algorithm gathers workspace connectivity information by performing a wavefront expansion (Sect. III-B). This expansion generates a tunnel of intersecting spheres S,



(a) (b)

Figure 2. A large box moves through a narrow maze: (a) focused EET search tree (b) Bridge-PRM with unnecessary exploration of areas that do not contribute to a solution path

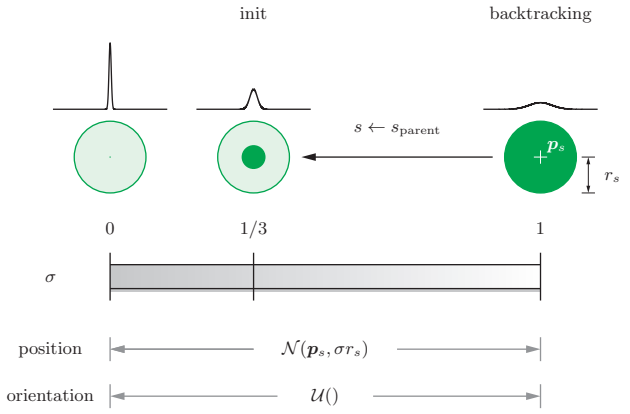


Figure 3. Influence of the σ value on the behavior of the EET planner: The variance of the normal distribution for the position is proportional to σ. The initial value of $\sigma = 1/3$ will place 99.7 % of the samples inside a workspace sphere. The orientation of these samples is then drawn from a uniform distribution. If σ reaches the maximum value of 1, the algorithm backtracks to the previous sphere and reinitializes σ.

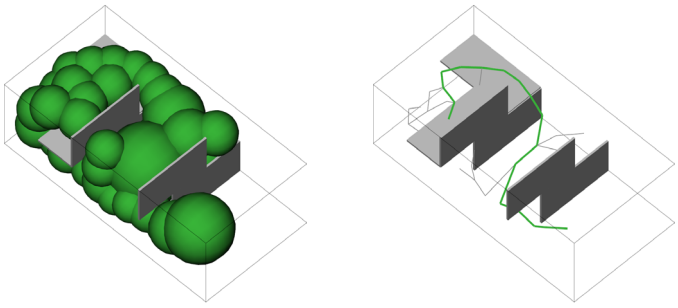
connecting start and goal in the robots workspace. The diameter of the individual spheres provides information about the environment's local geometric characteristics. The planner moves the robot along this workspace tunnel by "pulling" the robot's tool point toward the next sphere using the pseudo-inverse of the Jacobian matrix.

The behavior of the EET algorithm is driven by the value of a variable σ that balances exploration and exploitation. A value of $\sigma = 0$ indicates pure exploitation. Now, the planner behaves like a potential field planner based on an approximate global navigation function, represented by S [32]. When pure exploitation fails, σ starts to increase, leading to more exploratory behavior. Exploration is the result of sampling in task space from a Gaussian distribution around the centers of the spheres of S. The variance of this Gaussian is chosen to be proportional to σ (Fig. 3). The value of σ therefore indicates how closely the planner follows the workspace information contained in S when generating new samples.

To generate a novel sample from an existing one, the sample has to be translated and rotated. The translation is generated by sampling from a normal distribution around the center of the next sphere. The robot's end-effector orientation is sampled uniformly. When the value of σ exceeds 1, many samples would be placed outside of the spheres of S, indicating that the workspace information might not be helpful any longer and that the robot is "stuck". In this situation, the algorithm backtracks to the previous sphere and re-attempts tree expansion from a different configuration, effectively re-orienting and re-positioning the robot, in an attempt to get out of a local structural minimum.

B. Workspace Connectivity Information

The EET planner leverages several sources of information to perform exploitation and to balance between exploitation and exploration. Exploitation is performed based on global connectivity information for relevant portions of the workspace. This information is computed using a sphere-based wavefront expansion in workspace [32]. The wavefront expansion determines a tree of free workspace spheres (Fig. 4 and Fig. 5). Paths in the tree capture the connectivity of the free workspace and the size of the spheres along the path capture the amount of local free workspace. These spheres and their connectivity define an approximate workspace navigation function over parts of the workspace. This function will be used for exploitation and for guided exploration.



(a) (b)

Figure 4. Wavefront expansion with start position in the upper left and goal position in the lower right section: (a) distance computation in three-dimensional workspace provides information on narrow sections. (b) connectivity information is stored in a tree structure

More technically, the wavefront expansion incrementally grows a tree S where each node corresponds to a sphere. The expansion maintains a set of candidate spheres in a priority queue Q, in which spheres closer to the goal have a higher priority. In each step of the expansion, the highest priority sphere from Q is added to S. Whenever the algorithm moves a sphere s from Q to S, it enters new candidates spheres to Q. The new spheres are centered around points sampled on the surface of s. We set the size of the new spheres equal to the distance from its center to the nearest obstacle. This algorithm creates tunnels of free workspace. The expansion stops once one of these tunnels connects start and goal positions.

We extend the original method [32], which was applicable

Table I
SYMBOLS AND VARIABLES USED IN THE ALGORITHMS

Element	Description
α	control increase/decrease of exploitation (= 1 %)
γ	initial value for exploration/exploitation balance (= 1/3)
E	configuration space tree edge
ϵ	threshold for goal state comparison
G	configuration space tree
$\boldsymbol{J}$	Jacobian matrix
$\boldsymbol{p}$	workspace position of frame $\boldsymbol{T}$
Q	workspace sphere priority queue
$\boldsymbol{q}$	joint configuration
$\boldsymbol{R}$	workspace orientation of frame $\boldsymbol{T}$
r	workspace sphere radius
ρ	probability for sampling goal state in final sphere (= 50 %)
σ	control exploration/exploitation balance
s	workspace sphere
S	workspace sphere tree
$\boldsymbol{T}$	workspace frame with position $\boldsymbol{p}$ and orientation $\boldsymbol{R}$
V	configuration space tree vertex

Input: $\boldsymbol{p}_{\text{start}}, \boldsymbol{p}_{\text{goal}}, n$
Output: $S = (V, E)$
1: $V \leftarrow \emptyset$
2: $E \leftarrow \emptyset$
3: $Q \leftarrow \emptyset$
4: $r_s \leftarrow \text{DISTANCE}(\boldsymbol{p}_{\text{start}})$ *shortest distance to obstacles*
5: $s \leftarrow (\boldsymbol{p}_{\text{start}}, r_s, \varnothing)$ *sphere around start point*
6: $\text{INSERT}(Q, s, \|\boldsymbol{p}_{\text{goal}} - \boldsymbol{p}_{\text{start}}\| - r_s)$ *priority queue*
7: **repeat** *search until queue empty*
8: $s \equiv (\boldsymbol{p}_{\text{center}}, r, s_{\text{parent}}) \leftarrow \text{POP}(Q)$
9: $V \leftarrow V \cup \{s\}$
10: $E \leftarrow E \cup \{(s_{\text{parent}}, s)\}$
11: **if** $\|\boldsymbol{p}_{\text{goal}} - \boldsymbol{p}_{\text{center}}\| < r$ **then** *goal reached*
12: **return** S *sphere tree*
13: **end if**
14: $P \leftarrow \mathcal{U}(s, n)$ *sample n points on sphere surface*
15: **for** $\boldsymbol{p}_i \in P$ **do** *calculate sphere diameter for all points*
16: **for** $s_i \equiv (\boldsymbol{p}_{\text{center}_i}, r_i, s_{\text{parent}_i}) \in V$ **do** *spheres in tree*
17: **if** $\|\boldsymbol{p}_i - \boldsymbol{p}_{\text{center}_i}\| < r_i$ **then** *outside existing spheres*
18: $r' \leftarrow \text{DISTANCE}(\boldsymbol{p}_i)$ *distance to obstacles*
19: $s' \leftarrow (\boldsymbol{p}_i, r', s)$
20: $\text{INSERT}(Q, s', \|\boldsymbol{p}_{\text{goal}} - \boldsymbol{p}_i\| - r')$
21: **end if**
22: **end for**
23: **end for**
24: **until** $Q \equiv \emptyset$
25: **return** $\varnothing$

Figure 5. WAVEFRONT algorithm (comments in *gray* italics)

Input: $\boldsymbol{q}_{\text{start}}, \boldsymbol{q}_{\text{goal}}$
Output: $G = (V, E)$
1: $V \leftarrow \{\boldsymbol{q}_{\text{start}}\}$ *tree initialization with start configuration*
2: $E \leftarrow \emptyset$
3: $S \leftarrow \text{WAVEFRONT}(\boldsymbol{q}_{\text{start}}, \boldsymbol{q}_{\text{goal}})$ *compute sphere tree [32]*
4: $s \leftarrow (\boldsymbol{p}_s, r_s, s_{\text{parent}} \equiv S_{\text{begin}})$ *take first sphere*
5: $\sigma \leftarrow \gamma$ *initialize exploration/exploitation balance*
6: **repeat** *search until goal reached*
7: $(\boldsymbol{q}_{\text{near}}, \boldsymbol{T}_{\text{sample}}) \leftarrow \text{SAMPLE}(G, S, s, \sigma, \boldsymbol{q}_{\text{goal}})$ *sample new workspace frame and select nearest vertex in tree*
8: $\boldsymbol{q}_{\text{new}} \leftarrow \text{CONNECT}(\boldsymbol{q}_{\text{near}}, \boldsymbol{T}_{\text{sample}})$
9: **if** $\boldsymbol{q}_{\text{new}} \neq \varnothing$ **then**
10: $V \leftarrow V \cup \{\boldsymbol{q}_{\text{new}}\}$
11: $E \leftarrow E \cup \{(\boldsymbol{q}_{\text{near}}, \boldsymbol{q}_{\text{new}})\}$
12: $\sigma \leftarrow (1 - \alpha)\,\sigma$ *increase exploitation*
13: **for** $k \in S_{\text{end}} \rightarrow s$ **do** *search spheres backwards*
14: **if** $\|\boldsymbol{p}_{\text{new}} - \boldsymbol{p}_k\| < r_k$ **then** *position is within sphere*
15: $s \leftarrow k_{\text{child}}$ *advance to matching sphere*
16: $\sigma \leftarrow \gamma$ *reset exploration/exploitation balance*
17: **end if**
18: **end for**
19: **else** *expansion unsuccessful*
20: $\sigma \leftarrow (1 + \alpha)\,\sigma$ *decrease exploitation*
21: **end if**
22: **if** $\sigma > 1$ **then** *perform backtracking to previous sphere*
23: $s \leftarrow s_{\text{parent}}$ *select previous sphere*
24: $\sigma \leftarrow \gamma$ *reset exploration/exploitation balance*
25: **end if**
26: **until** $\|\boldsymbol{T}_{\text{goal}} - \boldsymbol{T}_{\text{new}}\| < \epsilon$
27: **return** G *configuration space tree*

Figure 6. EET algorithm (comments in *gray* italics)

only to free-flying robots. To address stationary and mobile manipulators, we force the tree expansion to first move from its start location toward the robot's base, prior to invoking the sphere expansion toward the goal location. This allows the end-effector to be retracted, should it be placed initially inside a narrow passage. For the mobile robots we require an additional free space tunnel from the start location of the base to its goal location.

C. Configuration Space Tree

The EET planner (Fig. 6) builds a configuration space tree, much like an RRT-based planner [10]. However, every vertex $\boldsymbol{q}$ in this tree G is associated with a corresponding workspace frame, consisting of the position and orientation of a control point on the robot. This point is the end-effector in case of articulated robots or an arbitrary point on the robot in case of rigid body robots.

The EET planner performs expansion of the configuration space tree G while balancing exploitation and exploration. Similar to RRT methods, a configuration is created toward which the tree should expand. Like the RRT-Connect algorithm [10], the EET applies a CONNECT step (Fig. 8) by executing multiple EXTEND steps (Fig. 9) until the robot collides or reaches a joint limit.

In contrast to RRTs, the direction of tree expansion is determined based on the workspace information contained in S. To achieve this, the tree of workspace spheres S is processed in depth-first fashion, considering only paths through the tree that lead to the specified goal location. The planner "pulls" the robot's control point into the direction indicated by the workspace connectivity information. This is accomplished by drawing a task frame from a range of distributions depending on the next sphere and σ in the SAMPLE function (Fig. 7), as described in Fig. 3. When the tree reaches the last sphere, the goal frame is chosen as extension direction with probability $\rho = 1/2$. After sampling, the algorithm determines the vector $\Delta\boldsymbol{x}$ pointing from the existing task frame toward the newly sampled frame (line 9.4). This displacement is translated into a displacement in configuration space using the pseudo-inverse of the Jacobian (line 9.5). The new configuration $\boldsymbol{q}_{\text{new}}$ is then tested for collision (line 9.6). To avoid numerical instabilities, the planner performs uniform random sampling in the vicinity of kinematic singularities (line 9.1). This sample moves the manipulator out of the singularity and the planner switches back to non-uniform sampling. If the sample is free of collision, it is added to the tree together with the corresponding edge (line 6.10). The value of σ is reduced, resulting in a shift toward exploitation (line 6.12). If the connection attempt fails, the value of σ is increased and the balance is shifted toward exploration (line 6.20). If the new task frame has entered a different sphere, the planner advances to the respective child sphere and resets the σ value to $1/3$ (line 6.16). If σ exceeds the upper limit of 1, the planner retreats to a previous sphere in the current workspace tunnel and also resets σ (line 6.23). The planner has already been able to create valid nodes for the previous sphere and will likely find new connections in the second run and continue to the next sphere.

D. Modifications of the EET Algorithm

The algorithm presented here contains several modifications of the original version [2]. The revised version includes a novel

Input: $G, S, s, \sigma, \boldsymbol{q}_{\text{goal}}$
Output: $\boldsymbol{q}_{\text{near}}, \boldsymbol{T}_{\text{sample}}$
if $s \equiv S_{\text{end}} \wedge \text{RAND}() < \rho$ **then** *within last sphere*
  $\boldsymbol{p}_{\text{sample}} \leftarrow \boldsymbol{p}_{\text{goal}}$ *select goal position*
  $\boldsymbol{R}_{\text{sample}} \leftarrow \boldsymbol{R}_{\text{goal}}$ *select goal orientation*
  $\boldsymbol{q}_{\text{near}} \leftarrow \text{NEAREST}(G, \boldsymbol{T}_{\text{sample}})$ *nearest vertex in tree*
else
  $\boldsymbol{p}_{\text{sample}} \leftarrow \mathcal{N}(\boldsymbol{p}_s, \sigma r_s)$ *sample within sphere*
  $\boldsymbol{R}_{\text{sample}} \leftarrow \mathcal{U}()$ *uniform sampling for orientation*
  $\boldsymbol{q}_{\text{near}} \leftarrow \text{NEAREST}(G, \boldsymbol{T}_{\text{sample}})$ *nearest vertex in tree*
end if
return $(\boldsymbol{q}_{\text{near}}, \boldsymbol{T}_{\text{sample}})$

Figure 7. SAMPLE algorithm (comments in *gray* italics)

Input: $\boldsymbol{q}_{\text{near}}, \boldsymbol{T}_{\text{sample}}$
Output: $\boldsymbol{q}_{\text{new}}$
$\boldsymbol{q}_{\text{new}} \leftarrow \varnothing$
repeat
  $\boldsymbol{q}_{\text{new}'} \leftarrow \text{EXTEND}(\boldsymbol{q}_{\text{near}}, \boldsymbol{T}_{\text{sample}})$
  if $\boldsymbol{q}_{\text{new}'} \neq \varnothing$ **then**
    $\boldsymbol{q}_{\text{near}} \leftarrow \boldsymbol{q}_{\text{new}}$
    $\boldsymbol{q}_{\text{new}} \leftarrow \boldsymbol{q}_{\text{new}'}$
  end if
until $\boldsymbol{q}_{\text{new}'} = \varnothing$
return $\boldsymbol{q}_{\text{new}}$

Figure 8. CONNECT algorithm (comments in *gray* italics)

method of sampling for the goal position in the last sphere to help reach the exact goal configuration. The revised version is also less prone to getting stuck during exploitative phases due to a novel sphere matching method and the associated backtracking through the sphere tree. The use of uniform sampling when the manipulator is near a singular configuration further improves the robustness of the planner. To further simplify the algorithm, the use of repulsive forces and normal distribution for orientation sampling has been removed.

IV. Experimental Results

In this section, we evaluate the performance of the proposed EET planner to demonstrate the importance of carefully balancing exploration and exploitation in motion planning.

A. Scenarios

We chose five experimental scenarios with varying characteristics to demonstrate the applicability of the EET planner to a wide range of application domains. All scenarios share the characteristic that useful workspace information is available for exploitation.

1) Piano Mover's Problem: In this classic motion planning scenario, a piano has to be moved from its current location in the upper left part of a $10\,\text{m} \times 10\,\text{m} \times 3\,\text{m}$ apartment (Fig. 10) to a different room in the upper right part. The apartment contains two narrow doorways, each requiring a coordinated translation and rotation of the piano.

2) Stationary Manipulator and Wall: A 6-DOF manipulator is mounted in front of a wall with four holes (Fig. 11). The robot starts with the end-effector inside the bottom left hole. The task is to retract the arm and reach into the bottom right hole. The initial and final configurations of this

Input: $\boldsymbol{q}_{\text{near}}, \boldsymbol{T}_{\text{sample}}$
Output: $\boldsymbol{q}_{\text{new}}$
if $\text{SINGULAR}(\boldsymbol{q}_{\text{near}})$ **then** *within singularity*
  $\boldsymbol{q}_{\text{new}} \leftarrow \text{RAND}()$ *uniform sampling for singularities*
else
  $\Delta\boldsymbol{x} \leftarrow \boldsymbol{T}_{\text{near}} - \boldsymbol{T}_{\text{sample}}$
  $\Delta\boldsymbol{q} \leftarrow \boldsymbol{J}^{\dagger}(\boldsymbol{q}_{\text{near}})\,\Delta\boldsymbol{x}$
  $\boldsymbol{q}_{\text{new}} \leftarrow \boldsymbol{q}_{\text{near}} + \Delta\boldsymbol{q}$
end if
if $\boldsymbol{q}_{\text{new}} \in \mathcal{C}_{\text{free}}$ **then**
  return $\boldsymbol{q}_{\text{new}}$
else
  return $\varnothing$
end if

Figure 9. EXTEND algorithm (comments in *gray* italics)

planning problem lie in difficult narrow passages and are near singularities.

3) Industrial Setting: A larger 6-DOF manipulator with a large, industrial gripper system has to perform an industrial pick and place motion. The robot has to move a large object from between two pillars to the top of a table (Fig. 12).

4) Mobile Manipulator: A holonomic 10-DOF mobile manipulator moves an L-shaped object in and out of L-shaped holes in a wall (Fig. 13). The two narrow passages in this scenario are very difficult, as workspace information is not helpful for aligning the object with the hole.

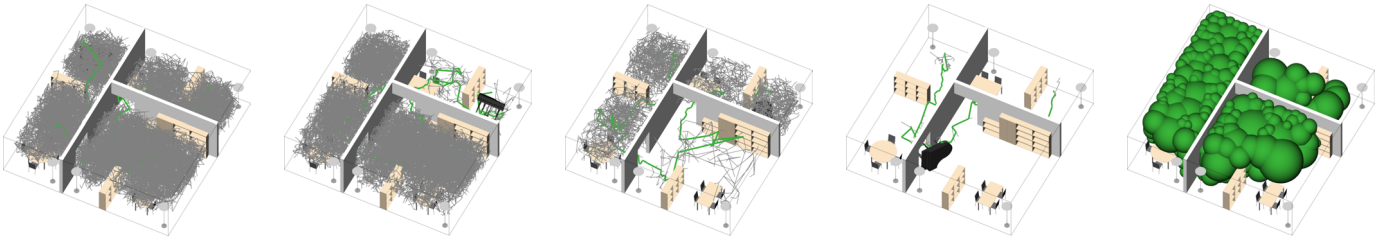
5) Protein-Ligand Interaction: This scenario is a motion planning problem inspired by protein-ligand interaction [38]. We model the ligand and the protein as spheres according to the van der Waals radii of the atoms. Ligand and protein have no internal degrees, but the ligand can translate and rotate, which leads to a 6-dimensional search space. The ligand has to find an exit pathway from the binding site inside of a protein (Fig. 14).

B. Experimental Setup

In our evaluation[1], we compare the proposed EET planner with a PRM planner with uniform sampling (referred to as PRM in the following sections) [7], PRM with Gaussian sampling (Gaussian-PRM) [11], PRM with bridge test (Bridge-PRM) [13], RRT-Connect with one tree (RRT-Connect1) and with two trees (RRT-Connect2) [10], as well as an adaptive dynamic-domain RRT (ADD-RRT) with two trees [21]. For these planners, a nearest neighbor search based on kd-trees [39] was used, with $k = 30$ for all PRM variants. To evaluate the impact of a task-space Voronoi bias on tree-based sampling, we also compare the proposed EET planner to a re-implementation of a task-space RRT planner described in the literature [34].

All algorithms have been implemented in C++ and are available in an open source implementation[2]. The implementation provides the means to recreate all experiments from Section IV. The benchmarks were run on a 2.27 GHz Xeon E5507 processor with 6 GB RAM and the Windows operating system. Collision tests and distance queries were performed

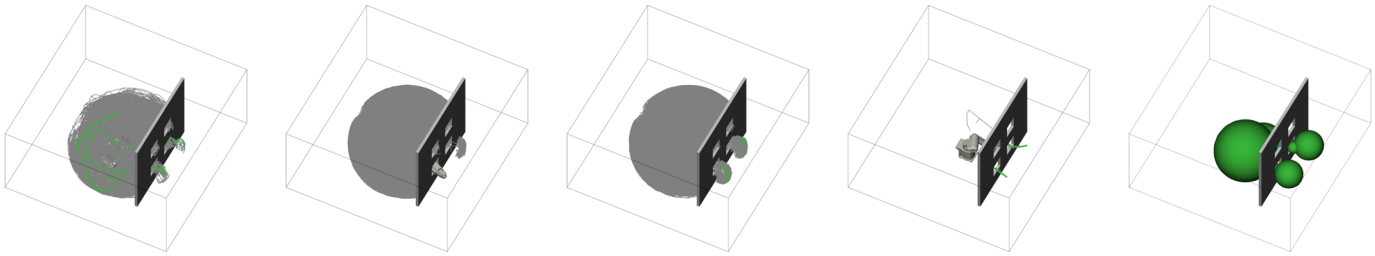
[1] youtube.com/playlist?list=PLcZc0GY2V0EzBq3fOJGFxyOmJO1Jeq1yL
[2] http://www.roboticslibrary.org/



(a) (b) (c) (d) (e)

Planner	Vertices	Collision Checks	Free (%)	Time (s)	Path Length	%
PRM	30 963±13 262	1 994 241± 742 180	87.27± 0.93	888.3±329.6	75.17± 12.11	65
Gaussian-PRM	21 412±13 358	1 616 636± 887 498	82.82± 1.73	700.4±391.1	84.19± 14.09	80
Bridge-PRM	19 623± 6 852	2 342 773± 747 079	54.24± 1.77	947.1±309.8	85.16± 12.13	50
RRT-Connect1	36 918±24 906	884 579± 506 446	79.43± 2.37	374.7±246.7	66.67± 7.33	95
RRT-Connect2	12 138± 7 251	337 360± 184 973	75.23± 3.70	154.3± 92.4	72.64± 8.29	100
ADD-RRT	11 032± 8 273	294 276± 201 838	80.34± 2.33	128.0± 97.2	65.66± 9.75	100
EET	398± 131	15 640± 6 210	43.30±10.43	16.6± 9.7	49.14± 7.58	100

Figure 10. The piano movers problem: (a) Bridge-PRM, (b) RRT-Connect1, (c) ADD-RRT, (d) EET, (e) wavefront expansion



(a) (b) (c) (d) (e)

Planner	Vertices	Collision Checks	Free (%)	Time (s)	Path Length	%
PRM	311 632± 8 929	15 632 505± 354 742	99.23± 0.01	1 200.0± 0.0	±	0
Gaussian-PRM	14 571± 5 685	1 645 463± 543 368	91.68± 0.47	119.9± 40.3	55.37± 23.10	100
Bridge-PRM	14 142± 6 494	2 089 098± 839 108	78.10± 1.35	155.3± 63.5	66.49± 15.83	100
RRT-Connect1	546 949± 36 234	16 672 822± 645 316	94.56± 2.50	1 200.0± 0.0	±	0
RRT-Connect2	323 098±159 403	10 836 882±4 733 644	85.82±11.11	831.6±352.6	53.94± 9.19	80
ADD-RRT	166 426±104 005	5 759 193±3 361 685	86.20±14.06	475.7±288.1	51.17± 12.60	100
EET	222± 108	3 236± 1 295	80.12± 6.94	1.3± 0.9	16.46± 6.33	100

Figure 11. Stationary manipulator reaching through openings in the wall: (a) Bridge-PRM, (b) RRT-Connect1, (c) ADD-RRT, (d) EET, (e) wavefront expansion

using the SOLID collision detection library [40] except for the protein benchmark. Due to the large number of spheres of the protein, we switched the collision detection library and used the more efficient broad-phase structure of Bullet[3] to speed up the queries.

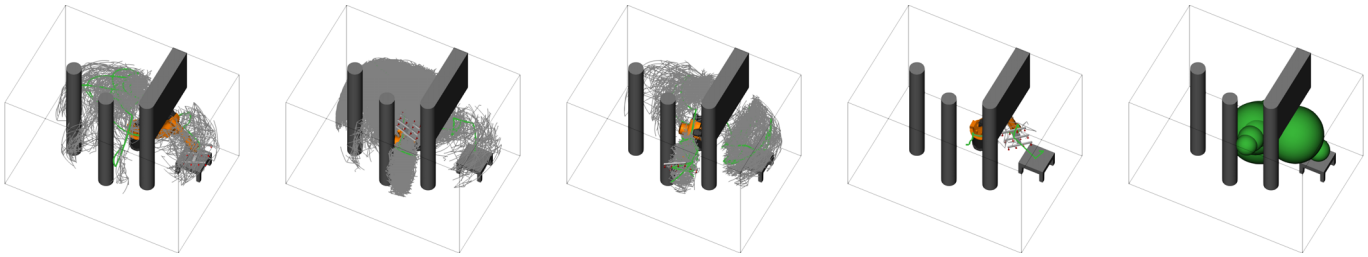
All results are averaged over 20 trials. If a planner was not able to solve a problem within 20 min, the experiment was aborted and considered as a failure. The tables in Fig. 10 to 14 show quantitative details of our experiments. Each table lists the number of vertices in the final graph, the total number of collision checks, the ratio of non-colliding versus colliding samples, the total run time until a solution is found, the length of the solution path (using the Euclidean norm), and the percentage of solved runs after 20 min of computation. For the EET planner, the computational cost of workspace information with the wavefront expansion is included in the total reported planning time. Average values are shown in black and standard deviations in light gray.

The images in Fig. 10 to 14 show one sample graph created by four different planners for each problem. Gray lines show the end-effector trajectories contained in the resulting tree or roadmap. These lines are a visual indication of the amount of configuration space exploration. A green line shows the final solution trajectory. The fifth image in each figure shows the workspace sphere expansion of the EET.

C. Evaluation

1) Planning Time: Our main objective in developing the EET planner was to achieve computationally efficient motion planning by balancing exploration and exploitation. The experimental results show that in all scenarios the EET planner is computationally more efficient than any of the planners we compare to. The EET planner achieves a success rate of 100 % for all scenarios, only matched by the ADD-RRT. Absolute

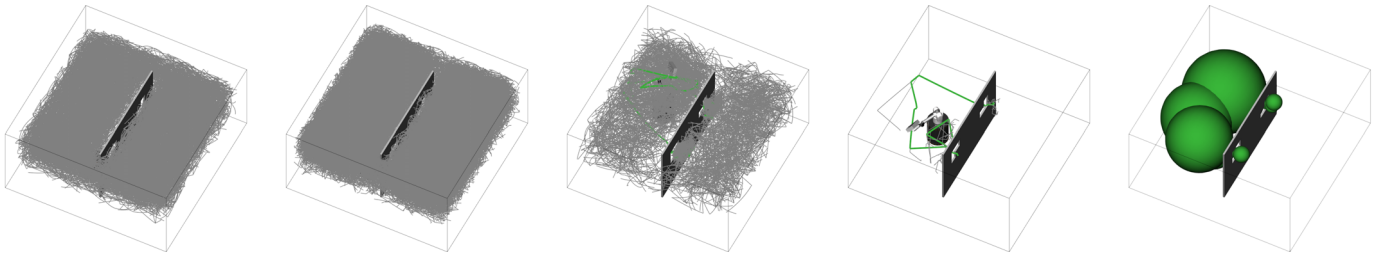
[3]http://www.bulletphysics.org/



(a) (b) (c) (d) (e)

Planner	Vertices	Collision Checks	Free (%)	Time (s)	Path Length	%
PRM	2 267± 1 774	305 164± 194 658	96.34± 0.30	174.8±103.9	65.07± 12.12	100
Gaussian-PRM	441± 222	107 294± 42 865	91.80± 0.91	70.2± 26.2	71.56± 22.17	100
Bridge-PRM	627± 329	156 619± 66 493	80.59± 1.81	98.9± 39.3	66.01± 21.10	100
RRT-Connect1	84 061±80 625	2 400 966±2 063 100	94.67± 0.28	499.0±451.5	45.63± 16.09	75
RRT-Connect2	2 922± 1 214	88 190± 42 627	86.57± 2.21	22.0± 10.1	48.29± 11.42	100
ADD-RRT	4 104± 2 196	125 091± 73 645	88.99± 1.67	29.6± 17.2	53.04± 11.17	100
EET	230± 171	8 045± 4 309	59.78±13.79	4.3± 3.0	21.09± 4.60	100

Figure 12. 6-DOF industrial manipulator with large gripper: (a) Bridge-PRM, (b) RRT-Connect1, (c) ADD-RRT, (d) EET, (e) wavefront expansion



(a) (b) (c) (d) (e)

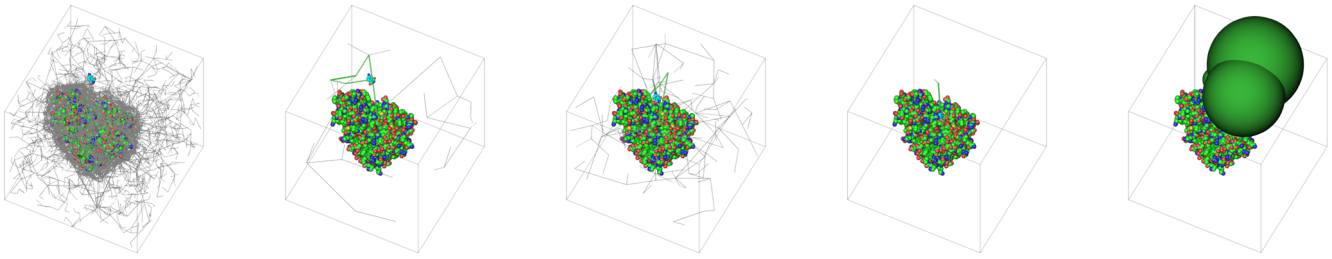
Planner	Vertices	Collision Checks	Free (%)	Time (s)	Path Length	%
PRM	32 649± 1 317	4 510 812± 170 073	98.94± 0.01	1 200.0± 0.0	±	0
Gaussian-PRM	29 402± 1 233	4 467 974± 167 851	96.17± 0.03	1 200.0± 0.0	±	0
Bridge-PRM	26 066± 875	4 500 360± 129 450	88.87± 0.07	1 200.0± 0.0	±	0
RRT-Connect1	93 745± 5 393	6 053 905± 317 765	97.47± 0.75	1 200.0± 0.0	±	0
RRT-Connect2	17 791±14 859	1 053 950± 961 849	88.00±15.67	203.9±184.6	49.89± 9.68	100
ADD-RRT	11 164± 6 346	520 089± 385 704	91.63± 4.56	105.3± 71.3	50.67± 11.34	100
TS-RRT	11 235± 3 081	475 058± 101 762	82.53±22.52	1 200.0± 0.0	±	0
EET	285± 161	12 732± 9 317	22.62± 7.42	15.8± 16.9	14.45± 4.04	100

Figure 13. L-shape attached to a mobile manipulator and corresponding wall openings: (a) Bridge-PRM, (b) RRT-Connect1, (c) ADD-RRT, (d) EET, (e) wavefront expansion

planning time is on the order of seconds for three of the five scenarios and about 20 s for the other two. The EET planner improves performance by at least a factor of 5.1. It achieves single-digit speedups in six cases, speed-ups of one order of magnitude in fifteen cases and two orders of magnitude in nine cases. Averaged over all scenarios and comparisons, the EET algorithm achieves a 152-fold speed-up.

Algorithmically, the EET planner is most similar to the RRT-Connect1 algorithm: Both compute a single configuration space tree. The positive effect of balancing exploration and exploitation can thus most easily be assessed by comparing the performance of the EET and the RRT-Connect1 planners. The speed-up of EET over RRT-Connect1 averaged over all five scenarios is 229-fold, i.e., two orders of magnitude; it varies from 8 in the protein/ligand scenario to over 900-fold in the stationary manipulator scenario. The low speed-up in the protein/ligand scenario is a result of the long, narrow, winding tunnel inside the protein the ligand has to move through. This tunnel prevents the RRT-Connect1 from unnecessarily exploring much of the free space. A RRT-Connect starting outside the protein performs several orders of magnitude worse. For the PRM, Gaussian-PRM, Bridge-PRM, and RRT-Connect2 planners the average speedups over the five experimental scenarios are 266-, 93-, 103-, 142-, and 79-fold, respectively (note that we aborted runs after 20 min).

For most scenarios, the ADD-RRT planner performs second best. It succeeds in all trials. However, there is a noticeable difference in performance in the stationary manipulator scenario. In this scenario, PRM-based methods outperform RRT-based planners (119.9 s vs. 475.7 s). We attribute this to the fact that the PRM's exploration strategy based on uniform random sampling is better suited for extended and curved narrow passages in configuration space. The EET planner outperforms all other planners in this scenario.



(a) (b) (c) (d) (e)

Planner	Vertices	Collision Checks	Free (%)	Time (s)	Path Length	%
PRM	63 148± 825	17 522 466± 196 153	99.96± 0.00	1 200.0± 0.0	±	0
Gaussian-PRM	11 709± 145	4 121 935± 45 526	97.39± 0.02	1 200.0± 0.0	±	0
Bridge-PRM	9 983± 137	4 975 107± 46 489	91.84± 0.08	1 200.1± 0.0	±	0
RRT-Connect1	201± 51	117 504± 93 360	39.86±27.26	44.1± 42.5	191.62±128.79	100
RRT-Connect2	33 323±23 611	7 031 273±4 071 890	99.04± 0.15	514.7±300.0	196.72± 72.84	95
ADD-RRT	1 626± 816	700 820± 262 954	99.57± 0.07	62.6± 22.6	200.61± 61.90	100
EET	108± 41	8 183± 2 987	76.88±17.65	5.0± 1.6	72.21± 23.45	100

Figure 14. Free-flying ligand and protein: (a) Bridge-PRM, (b) RRT-Connect1, (c) ADD-RRT, (d) EET, (e) wavefront expansion

The substantial performance improvements achieved by the EET planner are the result of effective use of information to balance exploration and exploitation. In all other respects, the algorithm is similar to RRT-Connect1. We will now further analyze the experimental results to provide more detailed support for this statement.

2) Performance Gains of the EET: We now analyze quantitatively which parts of the EET lead to the large decrease in planning time compared to simple RRT planners. In Table II we show the results of 10 planners. The first three planners employ a subset of the features the EET uses. The last seven planners are EET planner with varying parameters α and γ. The shown results are averaged over 20 executions for the mobile manipulation scenario from Fig. 13. Qualitatively, these results also hold true for the other scenarios.

The first planner we compare to for the sake of completeness is the single-tree RRT algorithm. As we already mentioned, this planner fails completely in this scenario.

Second, we want to see how much of the EETs performance can be attributed to task space sampling. Therefore, the second planner we compared to is a single-tree task-space RRT planner (TS-RRT) based on prior work [34]. This planner samples task space uniformly (position and orientation) to guide tree expansion and behaves like a RRT-Connect1 in every other aspect. Due to the high number of degrees of freedom, one might expect the performance of the TS-RRT to improve in the mobile manipulation scenario. The results in Fig. 13 show no clear advantage of task-space exploration in an RRT-based planner. The task-space Voronoi-bias guides the end-effector out of the first narrow passage but is unable to lead the planner to the entrance to the second narrow passage within 20 min. Without any additional information to guide sampling, it exhaustively explores the free space in front of the hole.

Table II
INFLUENCE OF THE PARAMETERS α AND γ IN THE SCENARIO WITH MOBILE MANIPULATOR SHOWN IN FIG. 13. THE PARAMETERS $\alpha = 0.01$ AND $\gamma = 0.3$ USED IN THIS PAPER ARE HIGHLIGHTED IN RED.

Planner	α	γ	Time (s)	%
RRT-Connect1			1 200.0	0
TS-RRT			1 200.0	0
Decomposition	0.0	0.8	826.0	40
EETα1	0.001	0.3	81.6	100
EETγ1	0.01	0.1	61.5	100
EET	0.01	0.3	15.8	100
EETγ2	0.01	0.6	20.2	100
EETγ3	0.01	0.9	90.1	95
EETα2	0.1	0.3	8.6	100
EETα3	0.3	0.3	252.3	80

The third planner we compare to is an EET variant that employs the same workspace information as the final EET but does not adaptively balance between exploration and exploitation. This planner is very similar to decomposition-based planning [32], although also applicable to stationary manipulators. We implemented this planner by setting the σ parameter of the EET to a constant value of 0.8. This value resulted in the best performance out of all constant σ values. Still this planner does not perform satisfactory, although it solves the problem within 1200 s in 40 % of the runs.

The last seven planners are equal to the EET planner as described in the previous sections with varying parameters α and γ. All but one of these variants perform better than the ADD-RRT—the second best planner from Fig. 13. Still the EET performs one order of magnitude better when the parameters are tuned. The planners EETγ1, EETγ2, and EETγ3 show how initialization of σ with the γ parameter influences the performance. The best performance is obtained for $\gamma = 0.3$.

The planners EETα1, EETα2, and EETα3 vary α, the degree of increase or decrease of σ. EETα2 shows the best performance with a relatively high value of $\alpha = 0.1$. This value lets σ rapidly oscillate between 0 and 1. The planner employs the backtracking step very often because σ quickly reaches the maximum value of 1. The very good performance shows that backtracking is a very powerful mechanism to

escape local minima. EETα3 shows that for a little higher value of $\alpha = 0.3$, the performance highly degrades and the planner does not find a solution at all in 20 % of the trials. In the failing runs, the planner gets stuck in a loop between backtracking and forward stepping of the workspace sphere tree. This endless loop does not happen for lower values of α, as the planner employs more local exploration and less backtracking. Therefore, we settled for a compromise between execution speed and risk of failure by setting σ to 0.01. In this case we never observed problems with the backtracking mechanism while still observing very good results.

3) Information Gathering: The acquisition of information requires computation. For the planner to gain efficiency, this cost must be offset by reduced computational cost during planning. In three scenarios, the sphere expansion requires less than 0.5 s—a small fraction of the overall planning time. In the protein/ligand scenario and the piano movers problem the high cost of collision checking and the large workspace causes this cost to increase to 1.5 s on average.

Our results show that the time expended on information acquisition is offset by computational gains in the planning phase in all scenarios.

4) Information Use: A planner that uses information effectively will need less samples to solve a planning problem. The EET planner required between 0.2 % and 10 % of the samples the next best planner needed to solve the five scenarios described here (individual data not shown). This supports our claim that the use of workspace information enables the EET planner to focus sampling on configuration space areas relevant to the solution.

The small number of placed samples also results in a lower number of vertices of the tree generated by the planner. For example, in the piano scenario, the RRT variants use about 27 times as many vertices as the EET planner; the PRM variants use about 49 times as many vertices.

Another way of interpreting this data is to say that the use of workspace connectivity information enables the EET planner to focus exploration on narrow passages, i.e., the most difficult areas of configuration space. Fig. 10(d) to 14(d) show few gray lines for the tree generated by the EET planner, most of them in the vicinity of narrow passages.

Fig. 15 shows the changes of the σ value during planning for the scenario with the stationary manipulator (Fig. 11).

5) Free Sample Placement: The sampling process of sampling-based motion planners, so our claim, can be guided beneficially by workspace information. If the workspace information is accurate, the EET planner should be able to generate the solution path by pure exploitation. In this case, we would expect the planner to generate very few colliding samples, because the workspace information indicates the correct expansion directions for the tree. When workspace information becomes less accurate or helpful, the planner must perform exploration to compensate for those inaccuracies. In this case, we would expect the planner to generate a higher fraction of colliding samples, used to acquire information about local configuration space obstacles. By balancing exploration and exploitation, the EET planner is able to gradually shift from one case to the other. We now analyze our results to show that the proposed planner does indeed possess this ability.

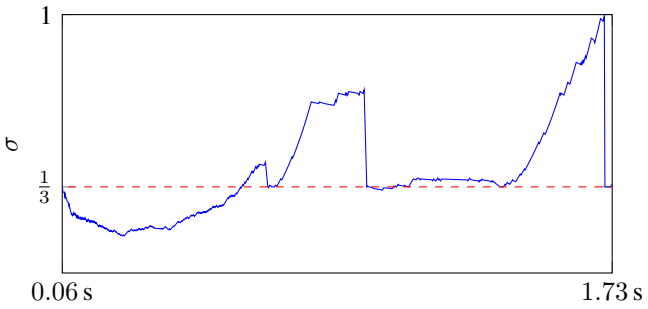


Figure 15. The solid *blue* line shows the progression of the σ variable in the stationary manipulator scenario (Fig. 11). After workspace exploration, the algorithm starts at 0.06 s with an initial value of $1/3$. Successful connections result in a balancing of the σ value, which is reset after advancing to the next sphere. In the last section, the arm has to enter the narrow passage which results in a lot of collisions and raises σ quickly. After backtracking once, the planner is able to find a solution path.

Let us first consider the case in which workspace information is highly accurate, such as the scenario shown in Fig. 2(a): A free-flying box with six degrees of freedom has to navigate through a narrow maze. Workspace information provides a strong clue about the correct configuration space direct for tree expansion. The EET planner is able to leverage this information to place 95 % of the samples in free space, resulting in a very simple configuration space tree. In contrast, the standard RRT planner places only 2 % of the samples in free space, spending most of the sampling effort on repeatedly running into walls. The PRM planners exhibit a different kind of pathological behavior in easy regions: They over-sample dramatically, as can be seen in Fig. 2(b). This means that they do not acquire useful information even though they do in fact place a high number of non-colliding samples. This comparison demonstrates that the EET planner is able to leverage high-quality workspace information very effectively.

In the second case, the planner should confine dense, collision-heavy sampling to difficult regions of configuration space. We just observed that PRM planners do not accomplish this. In our five scenarios, each of which contains narrow passages, the RRT and PRM variants place up to 99 % collision-free samples, performing unnecessary exploration in easy regions. At the same time, the rate of collision-free samples is almost always the lowest for the EET. Values range from 22.62 % in the mobile manipulation problem up to 80.12 % for the stationary manipulator. In some scenarios, one of the planners outperforms the EET planner in this category, indicating that the problem is particularly well-tailored to the planner's sampling strategy. However, a comparison of all scenarios shows that no single planner outperforms the EET in more than one scenario.

6) Path Length: Balancing exploration and exploitation also improves path quality. As quality measure we use path length in configuration space, as path lengths correlates with smoothness and execution time. In all five scenarios, the EET planner finds shorter paths than any of the other planners. This can be attributed to the use of workspace information to guide the extension of the configuration space tree.

The averages path generated by the other planners is more than 2.5 times longer than those generated by the EET planner. For the individual scenarios and planners, this number varies between 2.1 and 2.8. These substantial differences in path lengths result in very large subjective improvements of path quality during motion execution.

V. Limitations

While the EET planner has shown to be very efficient in practical scenarios, it remains—by design—incomplete and may fail under certain conditions. Failure might be caused by insufficient workspace information. For example, the workspace connectivity information might not capture an achievable path for the entire robot, either due to the robot's kinematic limitations or its geometric shape. However, the planner is in many scenarios still able to find a solution as it will explore beyond the sphere radii for high values of σ.

Fig. 16 shows an example where the workspace guidance information is misleading the EET. The robot has to rotate a long beam 180 degrees. To do so it needs to retract the red end of beam from the hole, rotate it, and insert the blue end of the beam into the hole. The EET cannot obtain useful workspace information for this problem. The workspace decomposition does not capture the needed rotation of the end-effector. In fact, the tunnel lets the robot initially insert the beam even more into the hole instead of retracting. This misleading information lets the EET perform two orders of magnitude worse compared two dual tree RRTs in this particular scenario. Note that the EET does not suffer from misleading information when the hole the beam is placed in initially is different from the target hole.

Fig. 17(a) shows another example: The workspace tunnel presented by the spheres generates a solution path around the target obstacle, rather than straight through the two vertical poles, as would be indicated by the workspace information. As a consequence, the planner requires more exploration but is still able to find a solution path. Fig. 17(b) shows a similar situation, in which the wavefront expansion generates a solution through the upper right hole in the wall rather than the lower right one.

As the degrees of freedom and the number of narrow passages increases, the situation changes: Fig. 17(c) shows a scenario with an 11-DOF manipulator in which the robot has to reach through two holes. In this case, the EET will fail because guiding the tool center point (TCP) of the manipulator alone is not sufficient to move through all holes of the wall. It should be noted, however, that this scenario is equally difficult for the other sampling-based planners we considered here. Switching between different TCPs during planning could provide a solution in this case.

Kinematic singularities do not affect completeness but limit the planner's ability to leverage information. Close to a kinematic singularity of the robot, the planner will generate very small successful expansion steps, causing the planner to generate more exploitative sampling. Such a scenario is shown in Fig. 17(d). To avoid this problem, we switch to uniform exploration in the proximity of singularities. Our algorithm finds a solution but wastes computation time by performing additional exploration. This problem could be addressed by recognizing this situation and adjusting σ accordingly.

There may also be other factors, in addition to kinematic singularities and structural local minima, that prevent the EET planner from reaching the exact goal configuration (see Fig. 17(e)). If failure occurs due to high local complexity of the planning problem around the goal configuration, a dual-tree version of the EET planner, with one tree expanding from the start configuration and one from the goal configuration, may be beneficial. However, this variant would require the specification of all joint angles, rather than just goal frame for the end-effector.

VI. Conclusion

We presented a computationally efficient, albeit incomplete, sampling-based motion planner. The planner outperforms standard sampling-based planners by up to three orders of magnitude on representative, challenging, real-world motion planning problems.

The proposed planner achieves its performance by deliberately balancing exploitation and exploration during planning. It acquires information about the planning problem in the low-dimensional workspace. This information captures some important structure inherent in the planning problem. The planner uses this information to guide configuration space sampling. When the information proves useful, the planner becomes increasingly exploitative. However, when the information does not lead to good planning progress, the planner gradually shifts its sampling strategy toward exploration. These shifts, as a function of the quality of available information, allow the planner to explore only small portions of relevant conformation space when helpful workspace information is available. Due to the planner's ability to shift between exploration and exploitation, we named the planner exploring/exploiting tree planner (EET planner).

The computational efficiency of the proposed EET planner comes at a cost. When the information used to guide configuration space sampling does not accurately capture the configuration space structure, the planner may fail or waste computational resources. We have analyzed and discussed several of these situations. We come to the conclusion that, in spite of these limitations, the EET planner is a highly efficient planner for complex, real-world planning problems.

References

[1] J. F. Canny, *Complexity of Robot Motion Planning*. MIT Press, 1988.

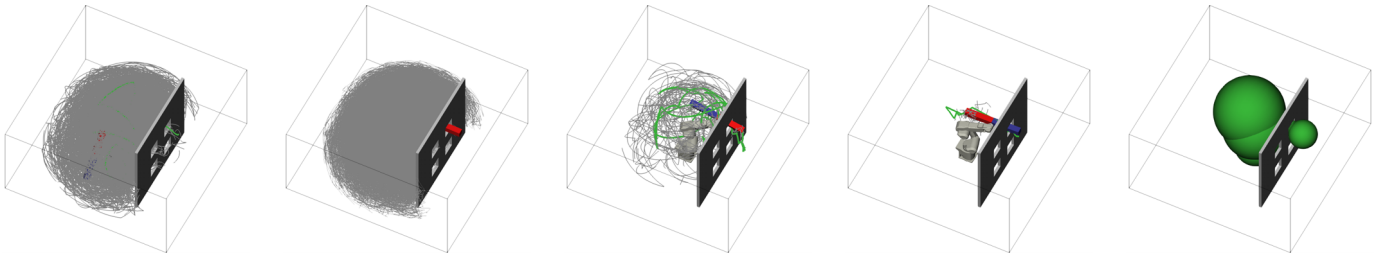
[2] M. Rickert, O. Brock, and A. Knoll, "Balancing exploration and exploitation in motion planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, May 2008, pp. 2812–2817.

[3] M. Rickert, "Efficient motion planning for intuitive task execution in modular manipulation systems," Dissertation, Technische Universität München, Munich, Germany, 2011.

[4] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, Mar. 1986.

[5] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, Oct. 1992.

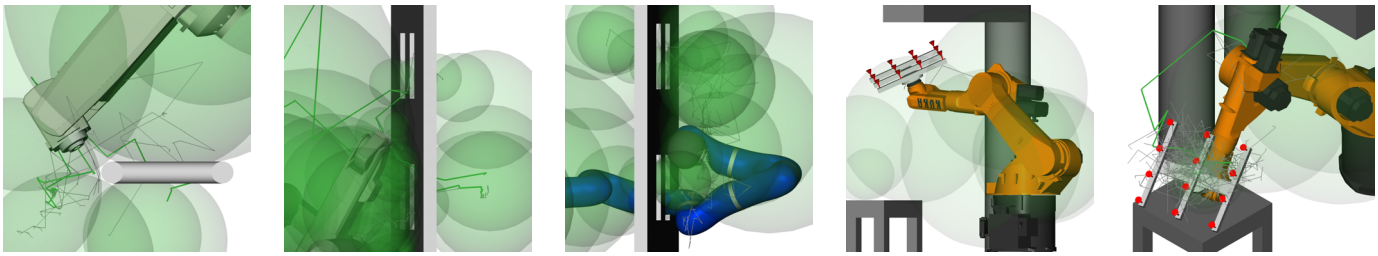
[6] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.



(a) (b) (c) (d) (e)

Planner	Vertices	Collision Checks	Free (%)	Time (s)	Path Length	%
PRM	311 632± 8 929	15 632 505± 354 742	99.23±0.01	1 200.0± 0.0	±	0
Gaussian-PRM	36 206± 40 507	3 045 096±2 877 873	88.04±2.15	335.3±318.3	44.64± 14.27	95
Bridge-PRM	27 679± 21 387	3 464 746±2 381 619	68.61±3.96	367.2±253.9	41.75± 17.27	100
RRT-Connect1	440 288±102 189	11 334 782±2 575 630	94.30±0.56	1 141.8±260.1	39.86±	5
RRT-Connect2	580± 409	31 916± 25 916	88.15±9.62	2.7± 2.2	27.33± 8.43	100
ADD-RRT	429± 234	16 637± 14 914	84.73±9.18	1.4± 1.2	29.05± 9.28	100
EET	1 134± 810	65 463± 40 512	10.07±2.80	217.8±265.2	24.97± 56.09	100

Figure 16. Stationary manipulator with large beam: (a) Bridge-PRM, (b) RRT-Connect1, (c) ADD-RRT, (d) EET, (e) wavefront expansion



(a) (b) (c) (d) (e)

Figure 17. Limitations of the proposed algorithm: (a) valid solution despite wrong workspace tunnel with goal configuration between cylindrical obstacles, (b) valid solution despite workspace tunnel through upper hole with goal configuration between lower hole, (c) failure due to large DOF and complex link between workspace and configuration space, (d) excessive exploration near singularity, (e) difficult to reach goal orientation due to use of only one tree.

[7] L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration space for path planning: Articulated robots," in *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, Munich, Germany, Sep. 1994, pp. 1764–1771.

[8] L. E. Kavraki, P. Švestka, 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, Aug. 1996.

[9] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Ames, IA, USA, Tech. Rep. TR 98-11, Oct. 1998.

[10] J. J. Kuffner, Jr. and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, USA, Apr. 2000, pp. 995–1001.

[11] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Detroit, MI, USA, May 1999, pp. 1018–1023.

[12] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*, Houston, TX, USA, Mar. 1998, pp. 155–168.

[13] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, Sep. 2003, pp. 4420–4426.

[14] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, Dec. 2000.

[15] D. Hsu, G. Sánchez-Ante, and Z. Sun, "Hybrid PRM sampling with a cost-sensitive adaptive strategy," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, Apr. 2005, pp. 3874–3880.

[16] S. Thomas, M. Morales, X. Tang, and N. M. Amato, "Biasing samplers to improve motion planning performance," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Rome, Italy, Apr. 2007, pp. 1625–1630.

[17] B. Burns and O. Brock, "Toward optimal configuration space sampling," in *Proceedings of the Robotics Science and Systems Conference*, Cambridge, MA, USA, Jun. 2005.

[18] J. Denny and N. M. Amato, "Toggle PRM: Simultaneous mapping of C-free and C-obstacle - a study in 2D -," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, USA, Sep. 2011, pp. 2632–2639.

[19] 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, Sep. 2012.

[20] A. Yershova, L. Jaillet, T. Siméon, and S. M. LaValle, "Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, Apr. 2005, pp. 3856–3861.

[21] L. Jaillet, A. Yershova, S. M. LaValle, and T. Siméon, "Adaptive tuning of the sampling domain for dynamic-domain RRTs," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Aug. 2005, pp. 2851–2856.

[22] J. Cortés, L. Jaillet, and T. Siméon, "Molecular disassembly with RRT-like algorithms," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Rome, Italy, Apr. 2007, pp. 3301–3306.

[23] B. Burns and O. Brock, "Single-query motion planning with utility-guided random trees," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Rome, Italy, Apr. 2007, pp. 3307–3312.

[24] S. Dalibard and J.-P. Laumond, "Linear dimensionality reduction in ran-

dom motion planning," *The International Journal of Robotics*, vol. 30, no. 12, pp. 1461–1476, Oct. 2011.

[25] M. Foskey, M. Garber, M. C. Lin, and D. Manocha, "A Voronoi-based hybrid motion planner," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, HI, USA, Oct. 2001, pp. 55–60.

[26] C. Holleman and L. E. Kavraki, "A framework for using the workspace medial axis in PRM planners," in *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, USA, Apr. 2000, pp. 1408–1413.

[27] C. Pisula, K. E. Hoff III, M. C. Lin, and D. Manocha, "Randomized path planning for a rigid body based on hardware accelerated Voronoi sampling," in *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*, Hanover, NH, USA, Mar. 2000.

[28] M. Garber and M. C. Lin, "Constraint-based motion planning using Voronoi diagrams," in *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*, Nice, France, Dec. 2002.

[29] J. P. van den Berg and M. H. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," *International Journal of Robotics Research*, vol. 24, no. 12, pp. 1055–1071, Dec. 2005.

[30] H. Kurniawati and D. Hsu, "Workspace-based connectivity oracle: An adaptive sampling strategy for PRM planning," in *Algorithmic Foundation of Robotics VII*, ser. Springer Tracts in Advanced Robotics. Springer, Apr. 2008, vol. 47, pp. 35–51.

[31] Y. Yang and O. Brock, "Adapting the sampling distribution in PRM planners based on an approximated medial axis," in *Proceedings of the IEEE International Conference on Robotics and Automation*, New Orleans, LA, USA, Apr. 2004, pp. 4405–4410.

[32] O. Brock and L. E. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 2001, pp. 1469–1474.

[33] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Discrete search leading continuous exploration for kinodynamic motion planning," in *Proceedings of the Robotics: Science and Systems Conference*, Atlanta, GA, USA, Jun. 2007, pp. 326–333.

[34] A. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space Voronoi bias," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2009, pp. 2061–2067.

[35] Y. Yang and O. Brock, "Efficient motion planning based on disassembly," in *Proceedings of the Robotics: Science and Systems Conference*, Cambridge, MA, USA, Jun. 2005, pp. 97–104.

[36] M. Stilman, "Task constrained motion planning in robot joint space," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, Oct. 2007, pp. 3074–3081.

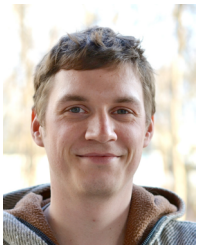
[37] J. M. Vandeweghe, D. Ferguson, and S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Pittsburgh, PA, USA, Nov. 2007, pp. 477–482.

[38] I. Al-Bluwi, T. Siméon, and J. Cortés, "Motion planning algorithms for molecular simulations: A survey," *Computer Science Review*, vol. 6, no. 4, pp. 125–143, Jul. 2012.

[39] A. Yershova and S. M. LaValle, "Improving motion planning algorithms by efficient nearest-neighbor searching," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 151–157, Feb. 2007.

[40] G. van den Bergen, *Collision Detection in Interactive 3D Environments*, ser. The Morgan Kaufmann Series in Interactive 3D Technology. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2004.

Markus Rickert is leading the Robotics research group at fortiss GmbH, An-Institut Technische Universität München in Munich, Germany. He received both his Diplom and doctorate in Computer Science from the Technische Universität München. His research interests include robotics, motion planning, human-robot interaction, cognitive systems, simulation/visualization, and software engineering.



Arne Sieverling is a Ph.D. candidate in Computer Science at the Technische Universität Berlin, Germany. He received the Bachelor's and a Master's degree in Computer Science in 2009 and 2011, both from the Georg-August Universität Göttingen, Germany. His research focuses on motion planning and motion generation for Mobile Manipulators.

Oliver Brock is the Alexander von Humboldt-Professor of Robotics in the Faculty of Electrical Engineering and Computer Science at the Technische Universität Berlin in Germany. He received his Diplom in Computer Science from the Technische Universität Berlin and his Master's and Ph.D. in Computer Science from Stanford University. He also held post-doctoral positions at Rice University and Stanford University. He was an Assistant Professor and Associate Professor in the Department of Computer Science at the University of Massachusetts Amherst, prior to moving back to the Technische Universität Berlin. The research of Brock's lab, the Robotics and Biology Laboratory, focuses on autonomous mobile manipulation, interactive perception, manipulation, and the application of algorithms and concepts from robotics to computational problems in structural molecular biology.