RRT*-Connect: Faster, Asymptotically Optimal Motion Planning

Conference Paper · December 2015

DOI: 10.1109/ROBIO.2015.7419012

7 authors, including:

Sebastian Klemm
FZI Forschungszentrum Informatik
18 PUBLICATIONS **111** CITATIONS
SEE PROFILE

Jan Oberländer
FZI Forschungszentrum Informatik
24 PUBLICATIONS **113** CITATIONS
SEE PROFILE

Andreas Hermann
FZI Forschungszentrum Informatik
35 PUBLICATIONS **117** CITATIONS
SEE PROFILE

Arne Roennau
FZI Forschungszentrum Informatik
88 PUBLICATIONS **447** CITATIONS
SEE PROFILE

 DOI: 10.1109/ROBIO.2015.7419012
IEEE International Conference on Robotics and Biomimetics (ROBIO), 2015, vol. 12, pp. 1670 - 1677, 6-9 Dec. 2015, Zhuhai, China.

RRT*-Connect: Faster, Asymptotically Optimal Motion Planning

Sebastian Klemm* Jan Oberländer* Andreas Hermann*
Arne Roennau* Thomas Schamm* J. Marius Zöllner* Rüdiger Dillmann†

***Abstract*— We present an efficient asymptotically-optimal randomized motion planning algorithm solving single-query path planning problems using a bidirectional search. The algorithm combines the benefits from the widely known algorithms RRT-Connect and RRT* and scores better than both by finding a solution faster than RRT*, and – unlike RRT-Connect – converging towards a theoretical optimum. We outline the proposed algorithm and proof its optimality. The efficiency and robustness is demonstrated in a number of real world applications which benefit from the bidirectional approach: planning car trajectories in a parking garage for the autonomous vehicle CoCar, generating cost-efficient trajectories for the multi-legged walking robot LAURON V in a planetary exploration scenario and performing mobile manipulation tasks for our highly actuated service robot HoLLiE. Moreover, we compare and show the improvements over "vanilla" RRT in a set of challenging benchmarks. RRT*-Connect will contribute to increase the performance of autonomous robots and vehicles due to the reduced motion planning time in complex environments.**

I. INTRODUCTION

Today, there exist many good approaches to solving navigation and manipulation tasks. However motion planning still is a crucial research topic. Although robotic hardware improves as new computers provide more computational capabilities, navigation and manipulation tasks still impose a challenge to be solved quickly and in a qualitatively optimal or at least sufficient manner. Scenarios where a planning algorithm has to find an optimal solution quickly while maintaining a certain quality level for the resulting paths are omnipresent in real-world environments. Typical pitfalls in these cases are narrow passages or maze-like arranged obstacles, where many motion planners struggle. Solving motion planning tasks that underly such constraints are challenging and need to be solved quickly while still guaranteeing optimal robot movement.

In this paper we present a single-query *bidirectional* planning approach for *optimal* motion planning, which outperforms state-of-the-art motion planning approaches. We combine the advantages from two well-known motion planning algorithms and show that the resulting motion planner finds solutions much faster while maintaining the property of converging to an optimal solution.

Our approach enhances the asymptotically-optimal RRT* algorithm [1] by combining it with the bidirectional na-

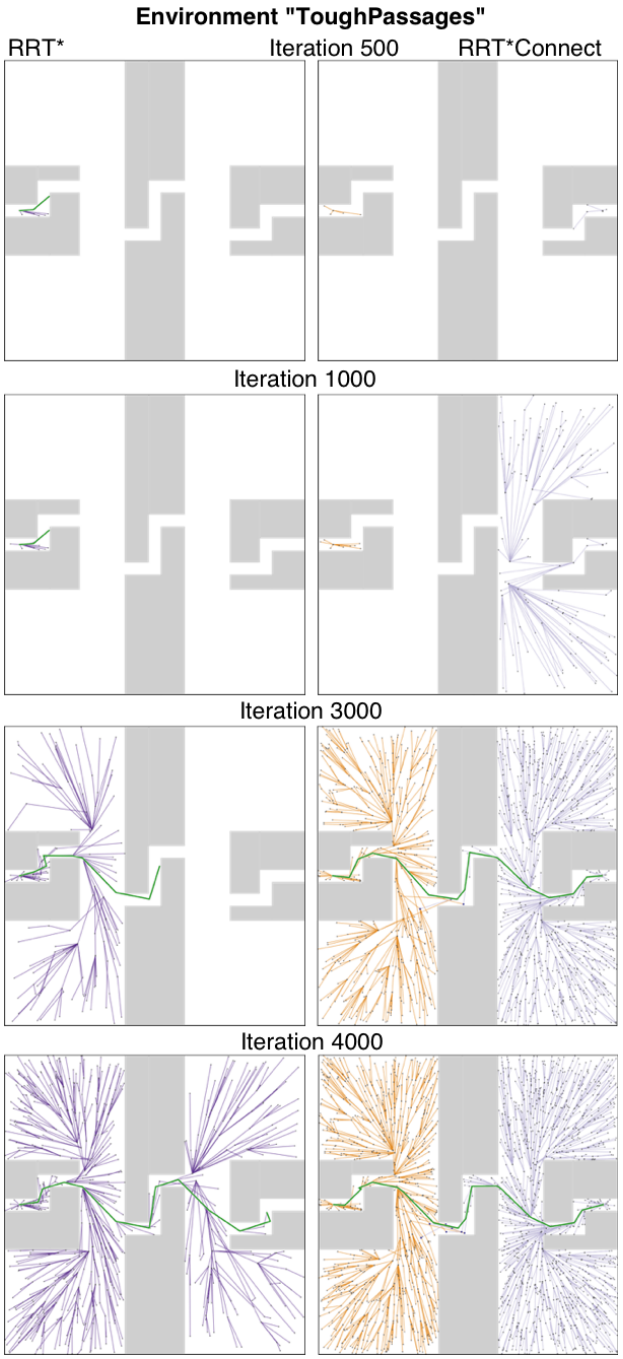

Fig. 1. Comparison of RRT* on the left and RRT*-Connect on the right hand side in planning iteration 500, 1000, 3000, 4000 on the same planning problem. In iteration 3000 RRT* has already found a solution (colored green) while RRT* has not even explored half of the statespace.

*Intelligent Systems and Product Engineering (ISPE); FZI Research Center for Information Technology, Haid-und-Neu-Straße 10–14, 76131 Karlsruhe, Germany
†Humanoids and Intelligence Systems Lab, Institute for Anthropomatics, Karlsruhe Institute of Technology.
{klemm, oberlaender, zoellner, dillmann}@fzi.de, dillmann@kit.edu

 DOI: 10.1109/ROBIO.2015.7419012
IEEE International Conference on Robotics and Biomimetics (ROBIO), 2015, vol. 12, pp. 1670 - 1677, 6-9 Dec. 2015, Zhuhai, China.

ture of RRT-Connect [2]. Bidirectional planning approaches prove very beneficial, especially in situations where the goal is difficult to reach: A unidirectional RRT planner has a relatively small chance of sampling a state that can reach a well-concealed goal. On the other hand, a bidirectional approach benefits from RRT's explorative character early on, performing many more attempts to grow the goal tree out of its concealed corner. A small but important change to the Connect routine of RRT-Connect makes it straightforward to show that our RRT*-Connect approach remains asymptotically optimal.

The remainder of this paper is organized as follows: In Section II we provide an overview of the motion planning problem and the research progress that has been made in the last decades up to today's state of the art. In Section III we discuss different sampling based RRT variants and present the functionality of the RRT*-Connect algorithm. Then we deduce some theoretical analysis regarding probabilistic completeness and asymptotic optimality of the presented algorithm. In Section V we provide evaluation results against other motion planning algorithms and show the merits of RRT*-Connect.

II. RELATED WORK

Motion planning is a research topic with applications in multiple fields, such as assembly analysis, virtual prototyping, computer animation, manufacturing, pharmaceutical drug design and robotics [3]. The motion planning problem can be described as the task to find a sequence of actions which, when applied to a given start state, will transfer the system to a given goal state, while respecting certain constraints, such as the requirement for the resulting path to be collision-free.

A subset of motion planning algorithms are the sampling based approaches, which try to solve the task using randomization and which are efficient for high-dimensional state spaces. Many of those algorithms provide probabilistical completeness (as defined later in Eq. 2). As a contrast, complete algorithms are also known [4] for this general class of problems, but the computational complexity limits their use to low-dimensional state spaces [5], e.g. the piano movers' problem has been shown to be PSPACE-hard [6]. This motivated to research in the field of path planning using randomization [7] [8]. The fundamental idea of randomized motion planning is to sample the state space instead of explicitly describing it, e.g. in a closed-form expression, which becomes harder or computationally infeasible with increasing dimensions of the state space and number of obstacles. Each sample is evaluated and classified into $\mathcal{X}_{\text{free}} \subseteq \mathcal{X}$, if the state is considered collision free, or else into $\mathcal{X}_{\text{occ}} = \mathcal{X} \setminus \mathcal{X}_{\text{free}}$, where $\mathcal{X}$ represents the state space of the system used for planning. More generally speaking, states are classified to be *valid* or *invalid*. With increasing number of sampling steps one can get an *impression* of how the state space looks like, where obstacles are present and where desired paths may be found. This allows some try-and-error attempts to find connections between the samples in $\mathcal{X}_{\text{free}}$. The result is a tree or roadmap, generally a graph, that approximatively models the connectivity within the state space, ideally containing a connection from start to goal states of the problem to be solved. The sampling can be done using different sampling strategies, e.g. using a uniform probability density function or such as described in [9] [10]. In [11] is shown how informed sampling strategies can be used to speed up *shortest path optimization* by sampling a heuristically derived subset of the state space.

In general the class of motion planners can be divided into single query and multiple query approaches. A single query approach tries to solve one planning problem, which is finding a valid path from a start to a goal state through the state space while respecting a variety of constraints, e.g. to be collision free. Such a solution of a planning problem is referred to as path, with the attribute *feasible* if all constraints are fulfilled. A multiple query approach tries to build a graph with underlying state space that may be used to solve multiple, slightly different planning problems, which is useful for spaces that have to be passed through very often.

The field of *probabilistic* motion planning experiences an increasing attention since the introduction of the single query Rapidly-Exploring Random Tree (RRT) motion planning algorithm, introduced by [12] in 1998. The RRT algorithm was designed to be capable of handling nonholonomic and kinodynamic constraints and high degrees of freedom [12] resulting in a high dimension of the state space. The RRT is, as already contained in its name, a randomized tree data structure that is built sequentially using a sampling procedure that features a high probability of expanding into unexplored regions of the state space. This is caused by the way how the algorithms divide the state space into subsets that suffice the Voronoi property [13], and the probability to sample into such a subset is proportional to the Lebesgue measure of that region [1]. The result is a quick exploration of the state space. The algorithm has been proven to be probabilistically complete [12], which means that with drawing enough samples, the probability that it finds an existing solution converges to one [14]. The algorithm is also fairly simple so it may be computed efficiently, even for high dimensional state spaces [12].

Since the introduction of the orignal RRT many varieties have emerged, often constructed for special subsets of planning problems and applications. Of particular note is the RRT-Connect approach [2], which incrementally builds two Rapidly-Exploring Random Trees originating at the start and goal states. Each of these trees explores the state space around them and advances towards its counterpart using a simple greedy heuristic, attempting to establish a connection between the trees and therefore solving the planning problem.

A variety of multiple query approaches exist, most importantly the Probabilistic Roadmap (PRM) approach, introduced by [15] in 1996. The PRM is a planner that constructs a roadmap of milestones. These milestones are states in $\mathcal{X}_{\text{free}}$. Each milestone is connected with a subset of its neighbors

 DOI: 10.1109/ROBIO.2015.7419012
IEEE International Conference on Robotics and Biomimetics (ROBIO), 2015, vol. 12, pp. 1670 - 1677, 6-9 Dec. 2015, Zhuhai, China.

if a connecting motion between them can be found. Solving multiple motion planning problems on the same state space can then be achieved by finding a connection between start and goal states towards the roadmap and a discrete search (such as e.g. A* [16]) on the roadmap.

The planning algorithms mentioned so far endeavor to *solve* a planning problem but do not provide any guarantees concerning the *quality* of a found solution. In 2014 Salzman et al. [17] presented a single query algorithm that is asymptotically-*near-optimal*, namely the LBT-RRT. In 2011, Karaman and Frazzoli [1] presented two variants of single (RRT*) and multi query (PRM*) motion planners, that guarantee *optimal* solutions of the planning problem, where optimality may be defined by e.g. the length of a solution path. This optimality is achieved by continuous resorting the tree (respectively roadmap) to always represent the minimum paths according to a given distance function.

Preliminary work on applying optimality properties onto a bidirectional search problem is available in the technical report [18], which provides some basic proof sketches on the probabilistic completeness and asymptotical optimality of such a planner.

The idea of our approach is to combine the cost optimality provided by the RRT* algorithm with the capabilities of the RRT-Connect approach to quickly find solutions. We provide a theoretical analysis on the probabilistic completeness and the asymptotic optimality of the algorithm as well as evaluation in *simulative scenarios* containing the well-known narrow-passage problems and mazes as well as *real world applications*: Navigating a multi-legged walking robot on rough terrain in a planetary exploration scenario, planning parking maneuvers for an autonomous car in an underground garage and generating whole body motions for a multi-degree of freedom service robot in a mobile manitpulation scenario. With the proposed RRT*-Connect algorithm we are able to extend the state of the art by an efficient and asymptotically-optimal motion planning approach that will contribute in speeding up a wide set of robotic applications.

III. RRT*-CONNECT

To depict the functionality of the presented RRT*-Connect, which inherits properties from other RRT-based algorithms, it is reasonable to start with a look at the basic RRT and RRT-Connect algorithms and then point out the relevant modifications and enhancements of the algorithms.

All RRT variants use one or more space filling trees [19] as the underlying data structure to store states and connections between those states. Such a tree is an undirected graph consisting of a set of vertices V representing states $x_i \in \mathcal{X}_{\text{free}}$ and a set of edges E connecting the vertices, without allowing cycles in the connections. For every RRT-related algorithm discussed in this paper the underlying tree is initalized containing one state x_{init}.

For the classic RRT algorithm this is the start state $x_{\text{init}} = x_{\text{start}}$. As listed in Alg. 2 the algorithm repeatedly samples a new state $x_{\text{rand}} \in \mathcal{X}$and then tries to extend the tree from the nearest state x_{nearest} already present in V towards x_{rand} using a steer function, resulting in x_{new} (see Alg. 1). This extension is examined considering validity regarding externally given constraints by what is usually called a local planner. If this validity is approved, x_{new} is added to V and an edge between x_{nearest} and x_{new} is created.

Algorithm 1: Extend

```
Function EXTEND(G = (V, E), x, x_new)
    // Extend G towards x, creating x_new
    x_nearest ← NEAREST(G = (V, E), x);
    x_new ← STEER(x_nearest, x);
    if OBSTACLEFREE(x_nearest, x_new) then
        V ← V ∪ {x_new};
        E ← E ∪ {(x_nearest, x_new)};
        if x_new = x then
            return REACHED;
        else
            return ADVANCED;
    return TRAPPED;
```

Algorithm 2: RRT

```
V ← {x_init}; E ← ∅;
for i = 1, ..., n do
    x_rand ← SAMPLEFREE_i;
    EXTEND(G = (V, E), x_rand, x_new);
return G = (V, E);
```

The RRT algorithm has been extended towards RRT-Connect using a bidirectional search. RRT-Connect (see Alg. 4) uses two separate trees T_a and T_b initialized respectively at the start and goal states, x_{start} and x_{goal} of the planning problem. A tree is extended just as in RRT, with the distinction that the algorithm alternates between the two trees. If an extend step was successful, the algorithm tries to connect the newly added state to the nearest neighbor of the other tree. For this connection attempt different strategies exist, see [2]. This behavior results in the trees T_a and T_b exploring their local surroundings and urging to connect with their counterpart, which leads to paths that are usually found more quickly than with a basic RRT, especially in scenarios where the start or goal states (or both) are concealed by barriers or narrow passages. This can be explained by the fact that the explorative strategy makes it easier to escape a small gap than to enter it. Additionally, the increased set of states each tree is trying to connect to provides a greater probability of finding a valid link to the other tree.

Algorithm 3: Connect

```
Function CONNECT(G_other = (V_other, E_other), x)
    // Repeatedly extend G_other towards x
    repeat
        S ← EXTEND(G_other = (V_other, E_other), x, x_new);
    until S ≠ ADVANCED;
    return S;
```

By always connecting the new sample x_{new} to the nearest neighbor $x_{\text{nearest}} \in V$, the RRT and RRT-Connect algorithms

 DOI: 10.1109/ROBIO.2015.7419012
IEEE International Conference on Robotics and Biomimetics (ROBIO), 2015, vol. 12, pp. 1670 - 1677, 6-9 Dec. 2015, Zhuhai, China.

Algorithm 4: RRT-Connect

$V_a \leftarrow \{x_{\text{init}}\}$; $E_a \leftarrow \emptyset$;
$V_b \leftarrow \{x_{\text{goal}}\}$; $E_b \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow \text{SAMPLEFREE}_i$;
  if $\text{EXTEND}(G_a = (V_a, E_a), x_{\text{rand}}, x_{\text{new}}) \neq \text{TRAPPED}$ **then**
    if $\text{CONNECT}(G_b = (V_b, E_b), x_{\text{new}}) = \text{REACHED}$ **then**
      return $G_a = (V_a, E_a), G_b = (V_b, E_b)$;
  $\text{SWAP}(G_a = (V_a, E_a), G_b = (V_b, E_b))$;
return $G_a = (V_a, E_a), G_b = (V_b, E_b)$;

only provide a local optimality property, neglecting that there might be a different state $\hat{x} \in V$ providing a better global path along the tree from x_{init} to x_{new}. This is where the approach by [1] comes in: the EXTEND step from Alg. 1 is modified to that shown in Alg. 5 and can be described as follows: As before, a new sample x_{new} is spawned and its nearest neighbor x_{nearest} already contained in the tree is determined. If the connection $(x_{\text{nearest}}, x_{\text{new}})$ is valid, x_{new} is added to V. But additionally, a *set* of nearest neighbors $X_{\text{near}} \subseteq V$ is inspected searching for a cheaper path to x_{new}. The connection representing the lowest cost is then added to the set of edges E. Further, the same neighborhood is checked for nodes that could be routed through x_{new} at an overall lower cost. This rewires the tree in a local neighborhood of every new tree node, improving overall costs. For better computational performance, the used set of nearest neighbors is limited to a subset of V (see Alg. 5). Karaman et al. [1] prove that if the neighborhood is chosen large enough (but by a function logarithmic in the number of vertices), the algorithm is asymptotically optimal [1, Theorems 38&39]. Using Alg. 5 as the extend strategy, the resulting RRT* algorithm as shown in Alg. 6 is not much different from the original RRT.

Combining the ideas of RRT-Connect and RRT*, then, is fairly straightforward: As with RRT-Connect, CONNECT* attempts to connect every new node x_{new} to the other tree, by extending the other tree towards x_{new}. Upon success, RRT-Connect stops, whereas RRT*-Connect will continue in order to find better solutions. This means that every node x_{new} which was successfully connected will be in *both* trees and therefore have two parents. Therefore the functions PARENT and COST require the relevant tree as an extra parameter, which would not be necessary for RRT*. The set $V_a \cap V_b$ therefore contains all nodes which connect the two trees.

RRT*-Connect (Alg. 8) mainly differs from RRT-Connect in that it does not terminate as soon as the first connection is found; instead it continues exploring the state space looking for better solutions. From the trees returned by Alg. 8, the best path is the one passing through the node

$$x_{\text{opt}} = \underset{x \in V_a \cap V_b}{\arg\min} (\text{COST}(x, G_a) + \text{COST}(x, G_b)) \ . \quad (1)$$

Algorithm 5: Extend*

Function $\text{EXTEND}^*(G = (V, E), x, x_{\text{new}})$
  // Extend G towards x with local optimization, creating x_{new}
  $x_{\text{nearest}} \leftarrow \text{NEAREST}(G = (V, E), x)$;
  $x_{\text{new}} \leftarrow \text{STEER}(x_{\text{nearest}}, x)$;
  if $\text{OBSTACLEFREE}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $V \leftarrow V \cup \{x_{\text{new}}\}$;
    $x_{\text{min}} \leftarrow x_{\text{nearest}}$;
    $X_{\text{near}} \leftarrow \text{NEAR}(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRT}^*}(\log(|V|)/|V|)^{1/d}, \eta\})$;
    $c_{\text{min}} \leftarrow \text{COST}(x_{\text{nearest}}, G) + c(\text{LINE}(x_{\text{nearest}}, x_{\text{new}}))$;
    // Connect along minimum-cost path among nearest neighbors
    foreach $x_{\text{near}} \in X_{\text{near}} \setminus x_{\text{nearest}}$ **do**
      if $\text{OBSTACLEFREE}(x_{\text{near}}, x_{\text{new}}) \wedge \text{COST}(x_{\text{near}}, G) + c(\text{LINE}(x_{\text{near}}, x_{\text{new}})) < c_{\text{min}}$ **then**
        $x_{\text{min}} \leftarrow x_{\text{near}}$;
        $c_{\text{min}} \leftarrow \text{COST}(x_{\text{near}}, G) + c(\text{LINE}(x_{\text{near}}, x_{\text{new}}))$;
    $E \leftarrow E \cup \{(x_{\text{min}}, x_{\text{new}})\}$;
    // Rewire the tree
    foreach $x_{\text{near}} \in X_{\text{near}} \setminus x_{\text{min}}$ **do**
      if $\text{OBSTACLEFREE}(x_{\text{new}}, x_{\text{near}}) \wedge \text{COST}(x_{\text{new}}, G) + c(\text{LINE}(x_{\text{new}}, x_{\text{near}})) < \text{COST}(x_{\text{near}}, G)$ **then**
        $x_{\text{parent}} \leftarrow \text{PARENT}(x_{\text{near}}, G)$;
        $E \leftarrow (E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\}$;
    if $x_{\text{new}} = x$ **then**
      return REACHED;
    else
      return ADVANCED;
  return TRAPPED;

Algorithm 6: RRT*

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow \text{SAMPLEFREE}_i$;
  $\text{EXTEND}^*(G = (V, E), x_{\text{rand}}, x_{\text{new}})$;
return $G = (V, E)$;

Algorithm 7: Connect*

Function $\text{CONNECT}^*(G_{\text{other}} = (V_{\text{other}}, E_{\text{other}}), x)$
  // Repeatedly extend G_{other} towards x
  repeat
    $S \leftarrow \text{EXTEND}^*(G_{\text{other}}, x, x_{\text{new}})$;
  until $S \neq \text{ADVANCED}$;
  // If $S = \text{REACHED}$, x will be in both trees
  return S;

Algorithm 8: RRT*-Connect

$V_a \leftarrow \{x_{\text{init}}\}$; $E_a \leftarrow \emptyset$;
$V_b \leftarrow \{x_{\text{goal}}\}$; $E_b \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow \text{SAMPLEFREE}_i$;
  if $\text{EXTEND}^*(G_a = (V_a, E_a), x_{\text{rand}}, x_{\text{new}}) \neq \text{TRAPPED}$ **then**
    $\text{CONNECT}^*(G_b = (V_b, E_b), x_{\text{new}})$;
  $\text{SWAP}(G_a = (V_a, E_a), G_b = (V_b, E_b))$;
return $G = (V_a \cup V_b, E_a \cup E_b)$;

 DOI: 10.1109/ROBIO.2015.7419012
IEEE International Conference on Robotics and Biomimetics (ROBIO), 2015, vol. 12, pp. 1670 - 1677, 6-9 Dec. 2015, Zhuhai, China.

IV. Analysis

A. Probabilistic completeness

A path planning problem considered *robustly feasible* if there exists a path with strong δ-clearance (at least a distance δ away from any obstacle) for some $\delta > 0$ [1]. An algorithm for solving any robustly feasible path planning problem ($\mathcal{X}_{\text{free}}$, x_{init}, $\mathcal{X}_{\text{goal}}$) is *probabilistically complete* if

$$\liminf_{n\to\infty} \mathbb{P}(\{\exists\ x_{\text{goal}} \in V \cap \mathcal{X}_{\text{goal}} \textit{ such that } x_{\text{init}} \textit{ is connected to } x_{\text{goal}} \textit{ in } G_n\}) = 1\ , \quad (2)$$

where G_n denotes the graph at planning iteration n and $\mathbb{P}(\,\cdot\,)$ is the probability measure [1]. In other words, the algorithm is probabilistically complete if, given enough time, it can find any solution to the planning problem, if one exists.

RRT*-Connect internally uses two distinct space filling trees that each fulfill the properties of classic Rapidly-exploring Random Trees [12]. Since the algorithm alternates between the two trees, growing each tree independently, both trees meet criterion (2). The CONNECT heuristic then brings the advantage that a feasible path may be found sooner, since the second tree effectively grows the goal region, speeding up the process of finding a connection between x_{init} and x_{goal}.

B. Asymptotic optimality

For a path planning problem ($\mathcal{X}_{\text{free}}$, x_{init}, $\mathcal{X}_{\text{goal}}$) and a cost function $c : \Sigma \to \mathbb{R}_{\geq 0}$, a solution path σ^* is called

a) *robustly optimal* if there exists a path σ' with strong δ-clearance in the same homotopy class [14], and $c(\sigma') = \min\{c(\sigma) : \sigma \text{ is feasible}\}$. The task of *optimal path planning* is to find such a robustly optimal path.

b) *asymptotically optimal* [1] if for any path planning problem and cost function that admits a robustly optimal solution,

$$\mathbb{P}(\{\limsup_{n\to\infty} Y_n = c^*\}) = 1\ , \quad (3)$$

where Y_n is the cost of the best path found up to planning step n and c^* is the theoretical optimal cost, $Y_n \geq c^*\ \forall\, n$.

RRT* has been proven to be asymptotically optimal in [20]. RRT*-Connect uses two RRT* trees, either of which remains asymptotically optimal. As the two trees grow independently and the state space samples are assumed to be independent and identically distributed, either tree independently converges to the optimal solution almost surely according to [20, Theorem 21]. Probabilistic completeness ensures that

$$\limsup_{n\to\infty} \mathbb{P}(\forall\, x \in V_{a,n}\ \exists\, x' \in V_{b,n} : x' \in \mathcal{B}_{x,\delta}) = 1\ , \quad (4)$$

where $V_{a,n}$ is the set of all vertices in G_a after n RRT*-Connect iterations, i. e., there will be nodes from both trees within a ball of radius δ from each other. In combination with the existence of a robustly optimal solution, this implies that the connect step will eventually succeed to link both trees along the optimal path. Finally the connect step results in these nodes being part of *both* trees. Since the convergence proofs do not rely on any special assumptions regarding nodes in $\mathcal{X}_{\text{goal}}$, the paths to any *inner* tree node also converge to their respective optima almost surely. Therefore, the connect step contributes towards finding an optimal solution.

In conclusion, we observe that RRT*-Connect maintains probabilistic completeness and speeds up the discovery of feasible paths, without the connect step having any negative effects on asymptotic optimality.

V. Evaluation

We evaluate RRT*-Connect against RRT* on multiple benchmark scenarios containing typical pitfalls ocurring in real world environments, such as narrow passages, barriers surrounding start and goal states and maze-like obstacle arrangements (see also Fig. 1 and 2). To generate comparable measurements we implemented the RRT*-Connect algorithm as part of the Open Motion Planning Library [21] and evaluate it against the contained implementation of RRT*. On each scenario we ran the planners 100 times, with 10 000 iterations on each planning algorithm. To make results comparable the planning problem is equal within each scenario, i.e. the start and goal states and all other parameters are identical, irrespective which planner is used or which benchmark run is performed.

A. Typical motion planning benchmark scenarios

Fig. 1 depicts a typical run of the two algorithms RRT* and RRT*-Connect trying to solve the same planning problem. As shown in the series of figures the RRT* algorithm finds a first solution to the planning problem more than 3 000 iterations sooner than RRT* (see also Tab. I). In Fig. 3 the median path lengths are plotted for both algorithms and different scenarios. The plots endorse the asymptotic optimality and show that RRT*-Connect provides solutions significantly faster in all scenarios. In some of our evaluation scenarios RRT* did not even find any solution within the given 10 000 iterations. The empirical results also indicate that while RRT*-Connect typically finds solutions much sooner, the overall convergence rate does not change for better or for worse.

Scenario	Algorithm	success rate	iteration of 1st solution 10th	50th	90th	median error
ToughPassages	RRT*	0.94	3059.0	5356.0	9372.0	0.4491
	RRT*-Connect	1.00	1490.5	2199.5	3439.5	0.4031
SquareFieldBW	RRT*	1.00	74.0	134.0	258.0	0.1338
	RRT*-Connect	1.00	20.0	40.0	79.0	0.1481
Maze	RRT*	0.21	6849.5	>10K	>10K	∞
	RRT*-Connect	1.00	1699.0	2713.0	4175.0	0.7431

TABLE I

COMPARISON OF RRT* AND RRT*-CONNECT ON TYPICAL MOTION PLANNING BENCHMARK SCENARIOS. A PLANNING PROCESS IS CONSIDERED TO BE SUCCESSFUL IF ANY SOLUTION FOR THE PLANNING PROBLEM WAS FOUND. THE ITERATION WHERE THE PLANNING PROBLEM FIRST FINDS A SOLUTION IS SHOWN WITH THE 10TH, 50TH (MEDIAN) AND 90TH PERCENTILES. THE MEDIAN ERROR OF THE BEST SOLUTION'S PATH LENGTH TO THE OPTIMAL SOLUTION IS LISTED IN THE LAST COLUMN.

 DOI: 10.1109/ROBIO.2015.7419012
IEEE International Conference on Robotics and Biomimetics (ROBIO), 2015, vol. 12, pp. 1670 - 1677, 6-9 Dec. 2015, Zhuhai, China.

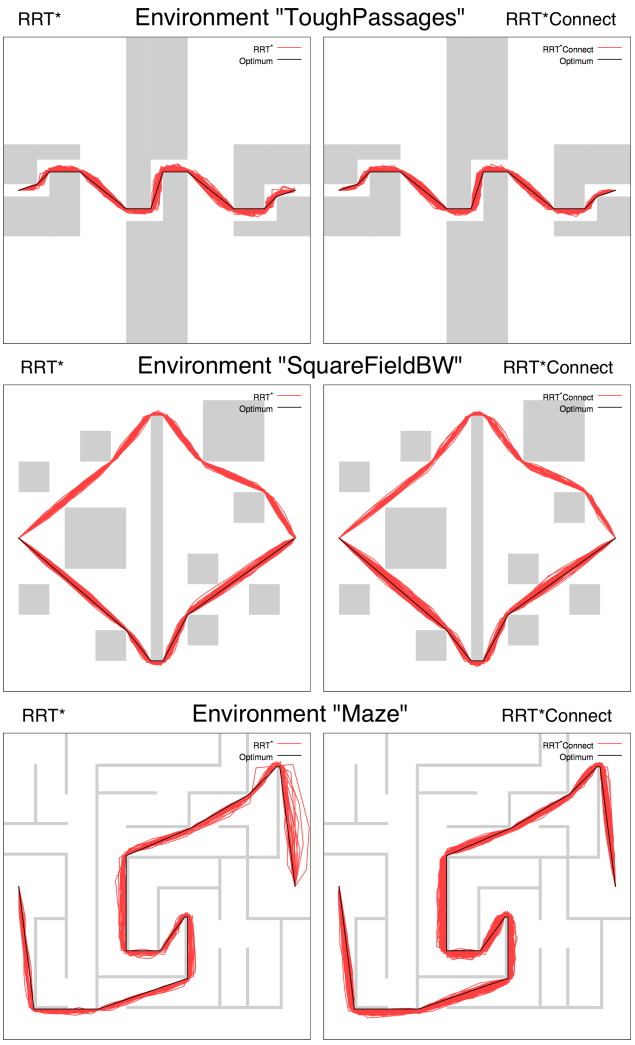

Fig. 2. Running RRT* and RRT*-Connect 100 times on different benchmark scenarios each allowing 10 000 iterations: Solution paths found by RRT* (left side) and RRT*-Connect (right side) are drawn in red. Allowing more iterations, the paths converge towards the theoretical optimum σ^*, shown in black. Notice that for the bottom row scenario RRT* has just recently found a solution, therefore not yet having found many *good* solutions near the optimum.

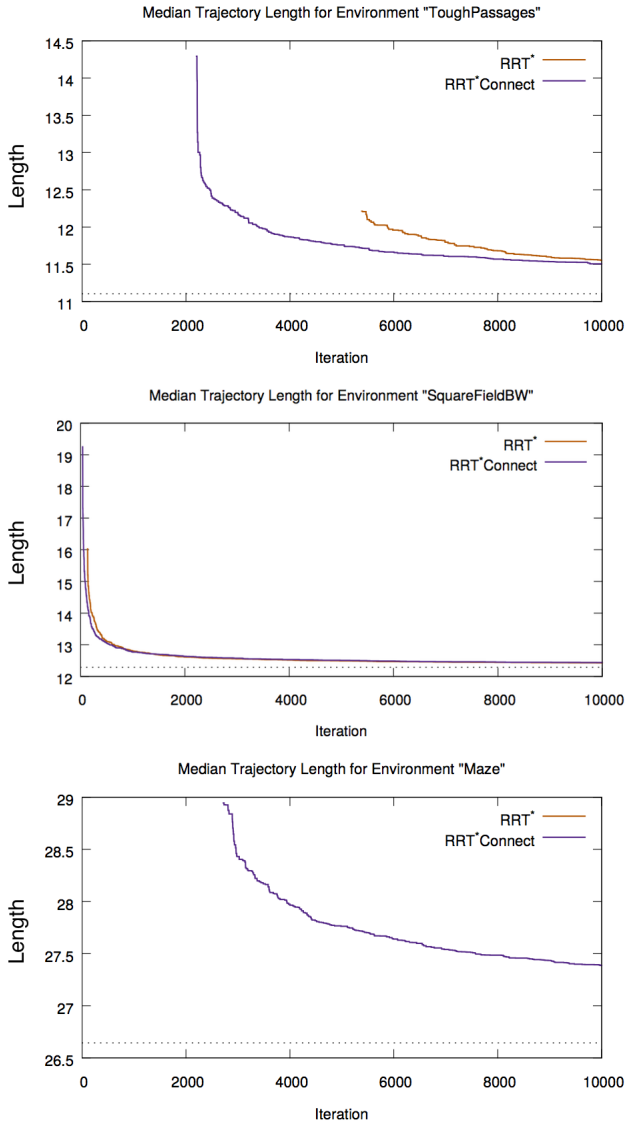

Fig. 3. Comparison of median trajectory lengths for RRT* and RRT*-Connect on different scenarios. The dotted lines in the plots indicate the length of the optimal path σ^*. It can be recognized that RRT*-Connect converges towards the theoretical optimum just like RRT*, with a similar convergence rate. But as is clearly visible, RRT*-Connect finds solution paths faster than RRT*. In the third graph those values differ too much to even be contained in the plots, i.e. RRT* does not find a solution within 10 000 iterations.

As announced before we further investigated the performace of the RRT*-Connect in real world application scenarios, such as planning for an autonomus vehicle in a parking garage and for a six-legged walking robot in a planetary exploration scenario.

B. Planning parking maneuvers

We utilized the RRT*-Connect motion planner to generate maneuvers for an autonomous car (such as shown in Fig. 4 (a)) navigating in a parking garage[1]. The underlying map is generated from laser scanner data. It stores truncated oriented signed distance functions (TOSDF) to obstacles' surfaces in a quad-tree like data structure (see Fig. 5). This allows an efficient collision checking by approximating the vehicle's footprint by a set of circles and checking the distances at the circle centers against their radii.

To generate parking trajectories, we use nearest neighbor calculus based on Reed-Shepp-curves [22], resulting in maneuvers suitable for confined spaces. The planning algorithm turned out to be well-suited for such maze-like environments with a lot of obstacles and boundaries. The constricted space around the final parking position, eventually surrounded by pillars, walls and other parking vehicles, presents a typical situation in which bidirectional search proves beneficial. The comparison of RRT* against RRT*-Connect is listed in the 1st row of Tab. II. It is obvious that the RRT*-Connect motion planner outperforms RRT* regarding planning time

[1] Video available at `https://youtu.be/G4XYMbtH758`

(a) (b)

Fig. 4. (a) CoCar – the instrumented cognitive car: a vehicle modified for autonomous driving, utilized to research in the fields of driver assistance systems, autonomous parking and autonomous driving on public road. CoCar is equipped with a variety of sensors, including multi-layer laser scanners. In our evaluation scenario the vehicle has to drive and park within an underground garage.
(b) Our six-legged walking robot LAURON V – biologically inspired by the Indian stick insect – with four degrees of freedom per leg. The robot is employed in search-and-rescue (SAR) and exploration scenarios on rough, unstructured and planetary-alike surfaces. Navigation in such scenarios requires optimal terrain traversal, highly adapted to the robot's stand stability, energy consumption and possible movement speed. Depending on the operation purpose the robot is equipped with a versatile sensor setup, including a rotating laser scanner for 3-D environment modelling.

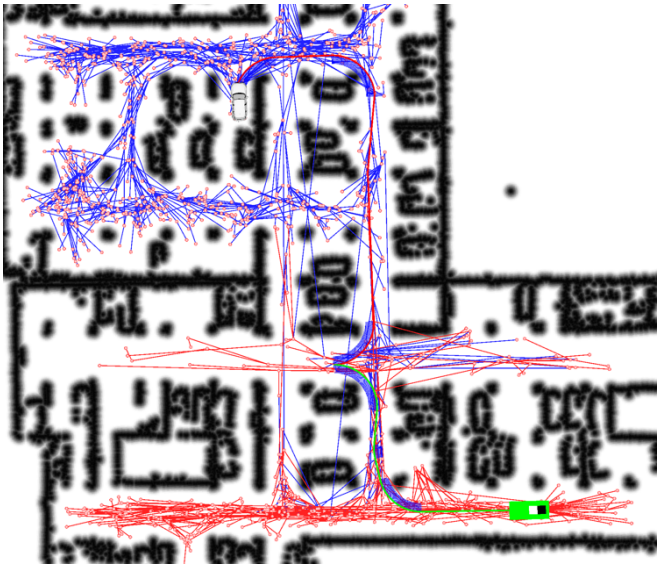

Fig. 5. Application scenario: Motion planning for an autonomous car in a parking garage. The RRT*-Connect algorithm is well-suited for finding paths in narrow environments. The vertices in the trees are indicated as red dots. The connection in between are edges in the tree. Actual motions obey the car's kinematics. The resulting trajectory is colored green for forward motion and red for backward motion.

and number of iterations to find a first solution for the planning problem.

C. Navigation in rough terrain

In [23] we introduced a navigation framework[2] that uses the RRT* planner to generate feasible and optimal motions for our six-legged walking robot LAURON V (shown in Fig. 4 (b)). The framework contains, inter alia, a multi-resolutional 2.5-D dimensional surface map (we refer to as *PlexMap*) generated out of 3-D laser scans, on which the planning steps are performed. The map's cells hold information about e.g. the terrains' roughness, surface normal and height of a specific area. Those parameters flow into a cost function, describing the desirability of the robot's configuration upon a specific cell for traversal. The cost function is crucial for the motion planner, as its evaluation is directly involved in the optimization process.

[2]Video available at `https://youtu.be/m8MXGFzjJxs`

Scenario	Algorithm	success rate	iteration of 1st solution 10th	50th	90th	time in [s]
OSDMap	RRT*	0.00	>10K	>10K	>10K	8.3
(Parking maneuvers)	RRT*-Connect	1.00	3016.0	5025.0	8119.2	3.8
PlexMap	RRT*	0.82	7681.3	9798.7	>10K	11.1
(Planetary exploration)	RRT*-Connect	1.00	5531.3	7551.1	9371.8	5.9
Cocktail bar	RRT*	1.00	5278.8	6878.2	8001.7	4.4
(Mobile manipulation)	RRT*-Connect	1.00	3821.7	4979.5	6102.6	3.2

TABLE II

COMPARISON OF RRT* AND RRT*-CONNECT ON REAL WORLD APPLICATION SCENARIOS. AS BEFORE, THE ITERATION WHERE THE PLANNING PROBLEM FIRST FINDS A SOLUTION IS SHOWN WITH THE 10TH, 50TH (MEDIAN) AND 90TH PERCENTILES. THE MEDIAN PLANNING TIME IN SECONDS (INCLUDING COLLISION CHECKING AND EVALUATION OF COST FUNCTIONS) REQUIRED TO FIND A FIRST SOLUTION IS LISTED IN THE LAST COLUMN.

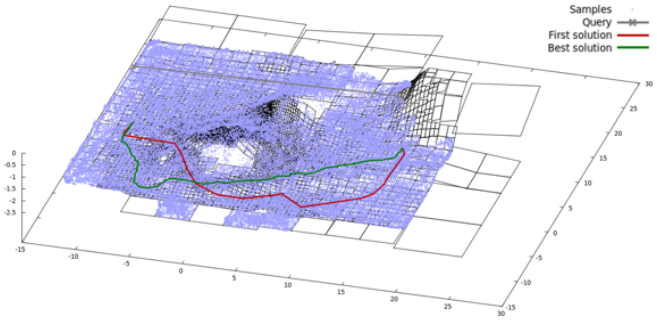

Fig. 6. Application scenario: Motion planning for rough terrain navigation. The figure shows the map on which the planning process is done. The blue points show sampled states during the planning process. The red line shows the first solution found by the planner, the green line shows the best solution regarding the utilized cost function.

We evaluated the RRT* motion planner on the planetary exploration scenario against the RRT*-Connect. The results are listed in the 2nd row of Tab. II.

The overhead of the RRT*-Connect algorithm performing Connect attempts is marginal in comparison to how much earlier solutions can be found in real world application scenarios. Therefore the total planning times required to find a first solution reduce heavily (see last column of Tab. II). Like in the previous evaulation scenario RRT* falls short and is outperformed by the RRT*-Connect algorithm.

D. Mobile manipulation with a service robot

We evaluated RRT*-Connect in a scenario where our service robot HoLLiE has to perform mobile manipulation tasks, i.e. mixing cocktails (see Fig. 7). We plan in a high dimensional state space: Three degrees of freedom (dof) for

 DOI: 10.1109/ROBIO.2015.7419012
IEEE International Conference on Robotics and Biomimetics (ROBIO), 2015, vol. 12, pp. 1670 - 1677, 6-9 Dec. 2015, Zhuhai, China.

Fig. 7. The service robot HoLLiE mixing cocktails. The scenario is used to evaulate RRT*-Connect in a complex mobile manipulation real world experiment. HoLLiE has to move its whole body between many obstacles.

the omnidirectional plaform, two dof for HoLLiE's active torso, which may be used to bend the robot, and each six dof for the left and right arm, resulting in 17 dof. The robot and its capabilities are explained in detail in [24]. Comparing RRT*-Connect against RRT* (see the last two rows of Tab. II) shows the reduced iterations and time needed to find a plan when using the proposed algorithm.

VI. CONCLUSIONS AND FUTURE WORK

We introduced an approach to combine the capability of quickly finding solutions even in tough planning scenarios from RRT-Connect with the guarantee to provide asymptotically optimal solutions to the planning problem like RRT*. We successfully demonstrated in several real experiments and benchmarking simulations that RRT*-Connect finds solutions earlier than the state of the art motion planner RRT* and also that convergence towards the optimal solution is provided, i.e. we showed that we could successfully combine all desireable properties of the former algorithms. This makes RRT*-Connect a valuable instrument for a large variety of planning scenarios.

In our future work we are going to investigate different variations of how the Connect step may be further improved, e.g. the heuristic approach from [2] looks promising for some scenarios. Further we are planning to contribute our implementation to the Open Motion Planning Library [21] and therefore provide access to RRT*-Connect.

ACKNOWLEDGMENT

This work was partially supported by the project AutoPLES under Federal Ministry of Education and Research (BMBF) grant 16N12348.

REFERENCES

[1] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[2] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An Efficient Approach to Single-query Path Planning," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 2000, pp. 995–1001.

[3] A. Bhatia and E. Frazzoli, *Incremental Search Methods for Reachability Analysis of Continuous and Hybrid Systems*. Springer Berlin Heidelberg, 2004.

[4] J. T. Schwartz and M. Sharir, "On the Piano Movers' Problem: Coordinating the Motion of Several Independent Bodies," *The International Journal of Robotics Research*, pp. 97–140, 1983.

[5] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA, USA: MIT Press, 1988.

[6] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proceedings of the IEEE Symposium on Foundations of Cumputer Science (FOCS)*, 1997.

[7] N. M. Amato and Y. Wu., "A Randomized Roadmap Method for Path and Manipulation Planning," in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, 1996.

[8] D. Hsu, J.-C. Latombe, and R. Motwani, "Path Planning in Expansive Configuration Spaces," *The International Journal of Computational Geometry and Applications*, 1997.

[9] J. Barraquand, L. Kavraki, J. Latombe, T. Li, R. Motwani, and P. Raghavan, "A Random Sampling Scheme for Path Planning," *The International Journal of Robotics Research*, pp. 759–774, 1997.

[10] V. Boor, M. Overmars, and A. van der Stappen, "The Gaussian Sampling Strategy for Probabilistic Roadmap Planners," in *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, 1999.

[11] J. D. Gammell, S. S. Srinivasa2, and T. D. Barfoot, "Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic," in *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[12] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," 1998. [Online]. Available: http://msl.cs.uiuc.edu/~lavalle/rrtpubs.html

[13] F. Aurenhammer, "Voronoi Diagrams – a Survey of a Fundamental Geometric Data Structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[14] S. M. LaValle, *Planning Algorithms*. Cambridge, UK: Cambridge University Press, 2006. [Online]. Available: http://planning.cs.uiuc.edu/

[15] L. Kavraki, P. Švestka, J.-C. Latombe, and M. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics and Automation*, p. 566–580, 1996.

[16] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in *4th Transactions on Systems Science and Cybernetics*, 1968.

[17] O. Salzman and D. Halperin, "Asymptotically near-optimal rrt for fast, high-quality, motion planning," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation*, 2014.

[18] M. Jordan and A. Perez, "Optimal Bidirectional Rapidly-Exploring Random Trees," Massachusetts Institute of Technology (MIT), Tech. Rep., 08 2013.

[19] J. J. Kuffner and S. M. LaValle, "Space-Filling Trees: A New Perspective on Incremental Search for Motion Planning," in *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[20] S. Karaman and E. Frazzoli, "Incremental Sampling-based Algorithms for Optimal Motion Planning," in *Proceedings of Robotics: Science and Systems II*, Philadelphia, PA, USA, 2006.

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

[22] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards." *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990. [Online]. Available: http://projecteuclid.org/euclid.pjm/1102645450

[23] J. Oberländer, S. Klemm, G. Heppner, A. Roennau, and R. Dillmann, "A Multi-Resolution 3-D Environment Model for Autonomous Planetary Exploration," in *IEEE International Conference on Automation Science and Engineering*, 2014, pp. 229–235.

[24] A. Hermann, J. Sun, Z. Xue, S. W. Rühl, J. Oberländer, A. Roennau, J. M. Zöllner, and R. Dillmann, "Hardware and Software Architecture of the Bimanual Mobile Manipulation Robot HoLLiE and its Actuated Upper Body," in *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Wollongong, NSW, Australia, July 2013, pp. 286–292.