Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Motion Planning with Cartesian Workspace Information ⋆

Bangshang Liu * **Christian Scheurer** ** **Klaus Janschek** *

* *Institut of Automation, Faculty of Electrical and Computer Engineering, Technische Universität Dresden, 01062 Dresden, Germany*
** *Department Corporate Research, KUKA Deutschland GmbH, Zugspitzstraße 140, 86165 Augsburg, Germany*

Abstract: We propose three extensions to the known sampling-based Exploring/Exploiting Tree (EET) Robot Motion Planner with following considerations: a) robot joint motion bounds, b) additional constraints on robot end-effector pose and c) parallelization of planning procedures to get alternative solutions. We also tackle the gap between global and local motion planning by combining sampling-based motion planning and reactive control approaches. These modifications complement the EET algorithm, which enables our planners to be more beneficial for practical applications. The experimental results demonstrate that our extended EET planners outperform other state-of-the-art sampling-based motion planners for some planning problems according to criteria such as planning time and path length.

Keywords: Sampling-based robot motion planning, Exploring/Exploiting Tree (EET), combination of global and local planners

1. INTRODUCTION

Motion planners can generally be divided into two categories: global and local motion planner LaValle (2006); Brock (2000). A *global motion planner* generates a sequence of robot motions from a start pose to a goal pose with full knowledge of the environment. This kind of planned motion shows planner's macroscopic understanding of the planning problem. In contrast, a *local motion planner* gathers local sensory informations to adjust the robot motion to dynamical and unpredictable changes in the vicinity of the robot Marín et al. (2018). Combining global and local motion planner into one framework offers beneficial combination of their individual strengths Brock (2000).

One main contribution of this paper is our SNS-integrated EET planner that combines the Exploring/Exploiting Tree (EET) planner Rickert et al. (2008, 2014) and the Saturation in the Null Space (SNS) algorithm Flacco et al. (2012, 2015). The Exploring/Exploiting Tree (EET) planner is a sampling-based motion planner. It trades deliberately probabilistic completeness for computational efficiency by using Cartesian workspace information and balancing exploration and exploitation while sampling Cartesian end-effector frames. The Saturation in the Null Space (SNS) algorithm aims to avoid exceeding robot joint motion bounds when tasks in workspace are to be accomplished. To achieve this, robot redundancy is used. According to our experimental results, this combination achieves better performance compared to other motion planners in some certain aspects.

⋆ This work was partly supported by the German Federal Ministry of Education and Research (BMBF) through the Hybr-iT project (grant no. 01IS16026A).

We also propose the Cartesian Constrained EET planner for additional restrictions on the robot end-effector pose. There are industrial applications where the end-effector is not allowed to operate in an arbitrary position and orientation. For example, the end-effector must be kept horizontally in pick-and-place applications, handling tasks or when palletizing packages in logistics industry. It is desired that a motion planner can take care of such constraints. Our third improvement of the EET planner tackles the incompleteness of the planner and covers more workspace information by running Parallelized EET planner which is explained in detail later on.

To the best of our knowledge, no such combined motion planner and variants of the EET planner have been proposed before. This paper is organized as follows. After providing a discussion of related work in the next section, our three variants of the EET planners are presented in Sect. 3. In Sect. 4 the performance of our planners is compared with those of other planners. The experimental results are analysed and evaluated. Finally, summary and conclusion are given in Sect. 5.

2. RELATED WORK AND CONTRIBUTIONS

2.1 Sampling-based Motion Planning

Sampling-based motion planning builds a road map (described as a graph) for solution paths which have to be in the free space of the configuration space LaValle (1998). Several motion planners have been developed and proposed in the past two decades.

The Constrained Bidirectional Rapidly-exploring Random Tree (CBiRRT) Berenson et al. (2009) handles a variety of constraints such as constraints on the pose of an object

held by a robot as well as torque limits. Projection technique is applied to explore the configuration space that corresponds to the predefined constraints in workspace. However, such projection results in large computational burden and low efficiency.
The Informed Rapidly-exploring Random Tree* (Informed RRT*) Gammell et al. (2014) extends the RRT* algorithm Karaman and Frazzoli (2010) by sampling informed in a configuration sub-space to improve convergence rate and quality of the final solution. However, the operation is only in configuration space instead of workspace. A further drawback is the difficulty of computing the configuration sub-space, especially in a high-dimensional one.
The Workspace Goal Regions (WGRs) presented in Berenson et al. (2009) describes a set of goal end-effector poses. However, solving the inverse kinematic is computationally costly and the WGRs focuses only bounds at the goal pose and not along the whole path.
Based on the decomposition-based motion planning framework Brock and Kavraki (2000) and sampling-based motion planning algorithms, the Exploring/Exploiting Tree (EET) Planner Rickert et al. (2008, 2014) presents free space represented by a sphere tunnel in the low-dimensional workspace and generates a path in the high-dimensional configuration space. The sampling scheme balances exploration and exploitation while sampling end-effector frames in the free space. This decomposed motion planning and unique sampling scheme improves significantly planning efficiency and solution quality. It achieves better planning performance than other sampling-based motion planners in most test cases Rickert et al. (2014). However, incompleteness and construction of one single sphere tunnel may leads to planning failure in some critical situations and limits its general application. Furthermore, no kinematical constraints, e.g. limits of joint velocity, joint acceleration, and constraints on the end-effector pose are taken into account.

2.2 Saturation in the Null Space (SNS)

The Saturation in the Null Space (SNS) technique Flacco et al. (2012, 2015) at the velocity level is a widely used redundancy resolution algorithm and is particularly investigated in our work. A kinematically redundant manipulator Siciliano (1990) can execute a task at the workspace level and meanwhile avoid singularities, joint limits and workspace obstacles Siciliano and Khatib (2016). As a local motion planner, the SNS algorithm strives to hold hard constraints of joint motion on position, velocity and acceleration while generating joint motion commands with a minimum norm property for a given velocity task. This method is meaningful and practical in robot applications. It makes full use of robot redundancy and uses task scaling to find iteratively and adaptively a feasible end-effector motion in workspace.

2.3 Combination of global and local motion planner

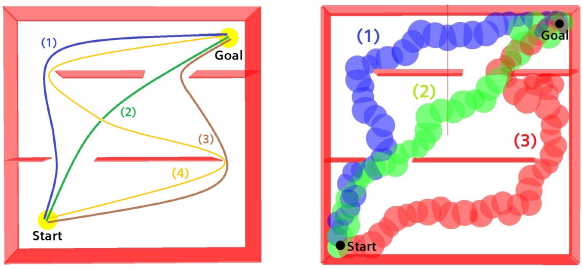
The motivation of combining global and local motion planner into one framework is explained in Brock (2000). The author proposed the elastic strip framework Brock and Khatib (2002), which is extended from the elastic-band framework Quinlan (1994) with several additional requirements, to accomplish such combination. The combined planner can not only react to local dynamical changes but also have a global view of the environment to generate an optimal path that is immune to local minima.
Our approach that combines global and local planners differs from the elastic strip framework by concerning joint motion limits and task preservation. Besides, no dynamic equation but inverse kinematics problem is taken into account in our proposed solution.

2.4 Parallelization of Motion Planning

The idea of applying multi-processing technologies to parallelize RRT motion planning is proposed in Devaurs et al. (2011); Jacobs et al. (2013). The authors in Devaurs et al. (2011) introduced an exploratory decomposition of building a single RRT. Each process performs parallelly its own sampling in the search space. Similarly, a Radial Subdivision Distributed RRT Jacobs et al. (2013) divides the configuration space into a few conical regions. In each sub-region the RRT algorithm searches the free space. Finally, these sub-trees are connected to form one single tree and a path solution is found.

2.5 Homotopy Classes of Paths

In some obstacle environments, there may be several feasible paths with different topological properties that are able to lead the robot from a start pose to a goal pose. Fig. 1a illustrates some alternative paths in a planning scene. These paths build a set of homotopy classes of paths Bhattacharya et al. (2010); Bhattacharya (2011). Among these path solutions one optimal path that has the least cost with respect to some criterion can be selected. It is desired that a robot motion planner is able to find out all the possible homotopy classes of paths and execute the best one.



(a) Paths with different homotopy

(b) Sphere tunnels with different homotopy

Fig. 1. Scene with possibility of parallelized planning

The contributions of our paper are modifications and optimization of the original EET planner to improve its performance in more general and practical planning scenarios, inspired by concepts and techniques mentioned above. We present three variants of the original EET planner that concentrates respectively on one requirement elucidated as follows.

- ***Joint motion bounds***: The robot joint motion bounds should not be exceeded throughout the whole motion along the planned path. For a redundant manipulator, these constraints can be guaranteed easily with the Saturation of Null Space (SNS) algorithm. It

is of benefit to integrate the SNS algorithm into the EET planner for strict joint motion limits especially for joint velocity and acceleration.

- ***Constraints on the end-effector pose***: In some practical planning problems, Cartesian constraints on the end-effector orientation should be taken into account. Taking advantage of the unique sampling scheme in the EET planner, such Cartesian constraints can be easily implemented.
- ***Multiple free space tunnels***: The original EET planner represents free space in one sphere tunnel, where subsequent planning is operated. Due to the incompleteness introduced by the decomposition-based motion planning, it is not guaranteed that this sphere tunnel can definitely lead to a successful planning procedure and a feasible Cartesian end-effector path. A feasible Cartesian end-effector path is understood as a path along which all robot joint motion is within its kinematic limitations and no collision between the robot body and obstacles occurs. In the worst case, the planner fails to find a path solution in this single tunnel and reports planning failure, even though one does exist in another tunnel which is not found by the original EET planner. Based on the concept of homotopy classes of paths, building several free space tunnels that possibly contain a feasible path and selecting the optimal one according to some criteria will increase the success rate of motion planning. Moreover, multiple sphere tunnels make parallelization of motion planning considerable.

Recently, a hierarchical path planner based on the RRT planer has been proposed in Mesesan et al. (2018). This hierarchical path planner shares many similar characteristics with the basic EET planner and our motion planners. First, it decomposes the motion planning problem in a global and a local sub-problem. In the global sub-problem the entire free workspace is partitioned to build a graph of disjoint polytopes that provides connection information. In the local sub-problem a task-space tree is constructed in a RRT-like way to explore each polytope, connect adjacent polytopes and ultimately reach the goal point. Second, while growing the task-space tree, end-effector frame is sampled instead of joint configurations. Balancing between exploitation in the current polytope region and exploration to the target region is implemented as well. Third, robot redundancy is used with help of the pseudo-inverse and nullspace of the jacobian matrix for obstacle avoidance. Forth, it parallelizes its local planning part. Beyond these peculiarities, our approach manages to deal with confined joint motion capabilities and constrained the end-effector pose. Our adaptive balancing between exploration and exploitation results in more efficient planning. Furthermore, the integrated SNS-algorithm promises further extensions to planning with stack of tasks.

3. EXTENDED EXPLORING/EXPLOITING TREE PLANNERS

Our three variants of the exploring/exploiting tree planners are intended to overcome shortcomings of the original EET planner. The *SNS-integrated EET planner* takes care of using robot redundancy and satisfying joint motion limits, especially velocity and acceleration limits. This integration demonstrates the benefits gained from combination of a global and a local planner. The *Cartesian Constrained EET planner* incorporates Cartesian constraints on the end-effector pose while planning. The *Parallelized EET planner* applies parallelization to generate multiple sphere tunnels and search trees to mitigate the incompleteness and increase the success rate of planning.

3.1 SNS-integrated EET Planner

The Saturation in the Null Space (SNS) algorithm solves the differential inverse kinematic problems, where hard joint limits of a redundant manipulator and a desired scalar task velocity v_{des} are given. During the joint velocity planning, the SNS algorithm exploits robot redundant degrees of freedom to avoid exceeding its joint motion bounds. When no redundancy is available any more, the desired task velocity is scaled down in order to still hold the joint restrictions. The EXTEND algorithm in Rickert et al. (2014) has the same functionality as the SNS algorithm. It computes a new joint configuration $\boldsymbol{q_{new}}$ that leads the end-effector pose towards a sampled end-effector frame $\boldsymbol{T_{sample}}$. The modified EXTEND algorithm applying the SNS method is shown in Algorithm 1.

Algorithm 1 EXTEND algorithm applying SNS

Input: $\boldsymbol{q_{near}}, \boldsymbol{T_{sameple}}, v_{des}, \delta t$
Output: $\boldsymbol{q_{new}}$
1: $\boldsymbol{T_{near}} \leftarrow$ ForwardKinematic($\boldsymbol{q_{near}}$)
2: $\boldsymbol{J} \leftarrow$ ComputeJacobianMatrix($\boldsymbol{q_{near}}$)
3: $\boldsymbol{\Delta x} \leftarrow (\boldsymbol{T_{sameple}} - \boldsymbol{T_{near}})$
4: $\dot{\boldsymbol{x}}_{\boldsymbol{task}} \leftarrow v_{des} \frac{\boldsymbol{\Delta x}}{\|\boldsymbol{\Delta x}\|}$
5: $\dot{\boldsymbol{q}}_{\boldsymbol{SNS}} \leftarrow$ SNSAlgorithm($\dot{\boldsymbol{x}}_{\boldsymbol{task}}, \boldsymbol{J}$)
6: $\boldsymbol{q_{new}} \leftarrow \boldsymbol{q_{near}} + \dot{\boldsymbol{q}}_{\boldsymbol{SNS}} \cdot \delta t$
7: **return** $\boldsymbol{q_{new}}$

The end-effector frame $\boldsymbol{T_{near}}$ and the Jacobian matrix $\boldsymbol{J}$ in configuration $\boldsymbol{q_{near}}$ are computed at first (line 1, 2). $\boldsymbol{\Delta x}$ denotes a 6×1 displacement vector from the nearest frame $\boldsymbol{T_{near}}$ in search tree to the $\boldsymbol{T_{sameple}}$ (line 3). A desired task velocity vector $\dot{\boldsymbol{x}}_{\boldsymbol{task}}$ is calculated according to $\boldsymbol{\Delta x}$ (line 4). Then, function *SNSAlgorithm(.)* given in Flacco et al. (2012) is called to compute a joint velocity $\dot{\boldsymbol{q}}_{\boldsymbol{SNS}}$ and a new joint configuration $\boldsymbol{q_{new}}$ is returned (line 5, 6). δt denotes step size of computing time.
The integration of the SNS algorithm in the EET planner exhibits following advantages:

(1) Joint motion limits for position, velocity and acceleration are considered during motion planning,
(2) Robot redundant degrees of freedom is sufficiently used to accomplish workspace task,
(3) Cartesian velocity task will be scaled down in critical situations where the desired velocity value v_{des} is too large regarding the robot motion limits and available redundancy. This down-scaling can be recovered adaptively to one afterwards, and,
(4) Given a task velocity, motion execution time for the robot can be estimated in advance after the path is generated.

3.2 Cartesian Constrained EET Planner

One of the most distinct features of the EET planner is sampling of Cartesian end-effector pose in the free sphere tunnel to extend its search tree. Modifying such sampling scheme permits additional restrictions on the end-effector pose. Consequently, the planned end-effector trajectory can not only avoid obstacles but also be amenable for certain constraints on its pose.
In the Cartesian Constrained EET Planner the SAMPLE algorithm in Rickert et al. (2008) is modified (Algorithm 2). Constraints on the end-effector pose are defined in a 6×2 matrix $\boldsymbol{C_{ee}}$ that bounds allowable translation and rotation with respect to a predefined reference frame $\boldsymbol{T_{ref}}$. The rotational constraints are given in Euler angles (ϕ, θ, ψ).

$$\boldsymbol{C_{ee}} = \begin{pmatrix} x_{min} & x_{max} \\ y_{min} & y_{max} \\ z_{min} & z_{max} \\ \phi_{min} & \phi_{max} \\ \theta_{min} & \theta_{max} \\ \psi_{min} & \psi_{max} \end{pmatrix}$$

Transformation matrix $\boldsymbol{T_{dev}}$ denotes deviation of the sampled pose $\boldsymbol{T_{sample}}$ from $\boldsymbol{T_{ref}}$. Position vector $\boldsymbol{p_{dev}}$ and rotation matrix $\boldsymbol{R_{dev}}$ of $\boldsymbol{T_{dev}}$ is sampled according to a normal distribution $\mathcal{N}(\boldsymbol{p_s}, \sigma r_s)$ and a uniform distribution $Rand(\boldsymbol{C_{ee}})$, respectively. $\boldsymbol{p_s}$ and r_s is centre point and radius of the current sphere $s_{current}$. σ is a variable balancing exploration and exploitation in the EET planner.

Algorithm 2 SAMPLE algorithm in Cartesian Constrained EET Planner

Input: $S, s_{current} \equiv (\boldsymbol{p_s}, r_s), \sigma, \boldsymbol{T_{goal}}, \boldsymbol{T_{ref}}, \boldsymbol{C_{ee}}$
Output: $\boldsymbol{T_{sameple}}$
1: **if** $s_{current} = s_{end}$ .AND. RAND() $< \rho$ **then**
2: $\quad \boldsymbol{T_{sameple}} \leftarrow \boldsymbol{T_{goal}}$
3: **else**
4: $\quad \boldsymbol{p_{dev}} \leftarrow \mathcal{N}(\boldsymbol{p_s}, \sigma r_s)$
5: $\quad \boldsymbol{R_{dev}} \leftarrow$ Rand$(\boldsymbol{C_{ee}})$
6: $\quad \boldsymbol{T_{dev}} \leftarrow \{\boldsymbol{p_{dev}}, \boldsymbol{R_{dev}}\}$
7: $\quad \boldsymbol{T_{sample}} \leftarrow \boldsymbol{T_{ref}} \boldsymbol{.} \boldsymbol{T_{dev}}$
8: **end if**
9: **return** $\boldsymbol{T_{sample}}$

If a frame is to be sampled in the last sphere s_{end} in the tunnel S, the goal frame $\boldsymbol{T_{goal}}$ may be taken as a sampled frame with a certain probability described by a scalar number ρ (line 1, 2). Otherwise, a frame $\boldsymbol{T_{sample}}$ is computed with $\boldsymbol{T_{ref}}$ and $\boldsymbol{C_{ee}}$ (line 4 - 7).

3.3 Parallelized EET Planner

As it is mentioned in section 2, several free space sphere tunnels and path solutions, if they exist in the planning scenario, can be found to compensate the incompleteness and increase the success rate of the planning. Subsequently, an optimal path solution according to some criterion can be selected to enhance the quality of a motion planning solution. The main focus in our parallelized EET Planner is not searching for all homotopy classes of paths in the free workspace, but, given these paths already found, the possibility of parallelizing the EET planning process. Under this assumption, two kinds of parallelization can be implemented in the EET planning:

(1) parallelizing the construction of sphere tunnels and,
(2) parallelizing the generation of exploring/exploiting tree in each sphere tunnel.

Fig. 1(a) shows a scenario with four homotopy classes of paths $\bar{\tau}_i, i = 1, 2, 3, 4$. Three sphere tunnels generated by the EET planner are depicted in Fig. 1(b). The generation of these tunnels along different paths are independent of each other and can thus be parallelized. It should be emphasized here that the presence of these three different sphere tunnels, i.e., Cartesian sub-workspaces, leads to the important fact that three configuration sub-spaces related respectively to these three tunnels arise implicitly. Finding a feasible end-effector path for a robot in one sphere tunnel is equivalent to finding a path in the related configuration sub-space. In our parallelized EET planner, each process can run the EET algorithm independently and parallelly in its own Cartesian sub-workspace and build a tree in its own configuration sub-space. As a result, several paths in configuration space can be found in these search tree and the best one can be selected regarding certain criteria, e.g. path length in joint space, path length in Cartesian space, no task scaling happening, etc.
Parallel construction of sphere tunnels is described in Algorithm 3.

Algorithm 3 Parallel Construction of Sphere Tunnels

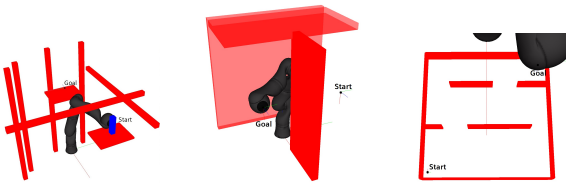
Input: $\boldsymbol{p_{start}}, \boldsymbol{p_{goal}}, \bar{\boldsymbol{\tau}} = \{\bar{\tau}_i, i = 1, ..., h\}$
Output: $\boldsymbol{S} = \{S_i, i = 1, ..., h\}$
1: **for** each process $i \in [1 : h]$ **do** parallely
2: $\quad S_i \leftarrow$WAVEFRONT$(\boldsymbol{p_{start}}, \boldsymbol{p_{goal}}, \bar{\tau}_i)$
3: **end for**
4: **return** $\boldsymbol{S}$

The difference between the extended function *WAVEFRONT(.)* and the original one in Rickert et al. (2008) is that there is not just one but several sphere tunnels $S_i, i = 1, ..., h$ that expand respectively along a given path $\bar{\tau}_i$. h is the number of paths $\bar{\tau}_i$.
The parallelized EET algorithm (Algorithm 4) is intended to plan one path solution in each sphere tunnel computed by Algorithm 3.

Algorithm 4 Parallelized EET Algorithm

Input: $\boldsymbol{q_{start}}, \boldsymbol{q_{goal}}, \bar{\boldsymbol{\tau}}$
Output: $\boldsymbol{\tau} = \{\tau_i, i = 1, ..., h\}$
1: $\boldsymbol{p_{start}} \leftarrow$ForwardKinematic$(\boldsymbol{q_{start}})$
2: $\boldsymbol{p_{goal}} \leftarrow$ForwardKinematic$(\boldsymbol{q_{goal}})$
3: $\boldsymbol{S} \leftarrow$Algorithm3$(\boldsymbol{p_{start}}, \boldsymbol{p_{goal}}, \bar{\boldsymbol{\tau}})$
4: **for** each process $i \in [1 : h]$ **do** parallely
5: $\quad G_i \leftarrow$EETAlgorithm$(\boldsymbol{q_{start}}, \boldsymbol{q_{goal}}, S_i)$
6: $\quad \tau_i \leftarrow$GetPath(G_i)
7: **end for**
8: **return** $\boldsymbol{\tau} = \{\tau_i, i = 1, ..., h\}$

Before the EET planning, different homotopy classes of paths $\bar{\boldsymbol{\tau}}$ are given and different sphere tunnels S_i are assigned to processors. Apart from S_i, all processors share the same planning parameters. During the expansion of its own tree structure G_i, each thread calls function *EETAlgorithm(.)* proposed in Rickert et al. (2008) (line 5). After all processors accomplish the tree construction, a set of paths $\boldsymbol{\tau}$ is returned (line 6, 8).



(a) Complex environment (b) Path around a wall (c) Labyrinth

Fig. 2. Three planning scenes for experimental results

4. EXPERIMENTS AND RESULTS

4.1 Scenarios

For performance evaluation, our motion planners shall accomplish motion planning tasks in the following three planning scenes (Fig. 2).
In scene (a), a KUKA-LBR 3 Albu-Schäeffer et al. (2007) robot shall carry a blue object from start to goal position in a complex environment where obstacles are shown in red. Collision between obstacles and robot body including the object is detected during the motion planning. In scene (b), the robot end-effector is expected to go around the wall to reach the goal pose in a confined workspace. Both planning problems above are three-dimensional. Test environment (c) is a two-dimensional labyrinth with relatively small range in height. There are several homotopy classes of paths in this labyrinth.

4.2 Experimental Setup

As a programming environment, the Microsoft Visual Studio Express 2015 for Desktop in Windows 7 operating system is used. The laptop is Dell Latitude E6520 with Intel® Core i5 2540M, 2 CPU cores with 4 threads, 2.60 GHz, max. 3.3 GHz.

4.3 Evaluation

Criteria for evaluating planners' performance are defined and explained here.

- *Planning time* represents the time duration one planner needs to find a feasible path solution for a planning problem. It is one of the most significant value that describes how fast and how well a motion planner performs.
- *Number of samples* indicates how many samples a motion planner made to solve the problem. It exhibits planning efficiency.
- *Satisfy rate of collision detection* shows capability of a planner to explore the collision-free space. High satisfy rate implies that the planner searches mostly in collision-free workspace and has high possibility to find a collision-free path. Fewer times of collision detection also induce shorter planning time because collision checking is the most time-consuming operation during planning.
- *Success rate of extend operation* shows the extend efficiency. Higher success rate indicates that a planner is more capable to generate a feasible joint motion to extend the search tree.
- *Number of tree nodes* describes the demand on computer memory. More nodes in the configuration space tree means more memory consumption.
- *Path length in euclidean space* exhibits length of the planned path in workspace, according to Euclidean distance.
- *Path length in configuration space* is the path length in the robot joint space, according to Euclidean distance.

SNS-integrated EET Planner Performance of the SNS-integrated EET planner and the basic EET planner is evaluated in the planning scenes (b). Both planners have configurations defined in the Appendix. Each planning runs successively 50 times and the average values of the planning results in Table 1 are evaluated.

The experimental results demonstrate that the SNS-integrated EET planner outperforms the basic EET planner due to following reasons:

(1) The SNS-integrated EET planner has higher sampling and extending efficiency. Because, when robot joint motion bounds are exceeded, the SNS algorithm tries to use robot redundancy and even to scale down the desired velocity task to still accomplish the extending operation, while the basic EET planner returns failed extending operation and samples a new frame, which results in low sampling and extending efficiency.
(2) The SNS-integrated EET planner achieves over 20% reduction of planning time and fewer number of tree nodes than the original EET planner.
(3) With fewer samples, the EET planner with SNS algorithm reduces collision checking times and has a higher satisfy rate of collision detection.
(4) The path length in both workspace and configuration space become shorter with the use of the SNS algorithm.

However, integrating the SNS algorithm in the EET planner enhances the code complexity and requires additional computation. Besides, kinematical redundancy of a robot is required.

Cartesian Constrained EET Planner Our Cartesian Constrained EET planner is investigated in comparison with the Constrained Bi-directional RRT (CBiRRT) planner. Both planners focus on additional restrictions on the end-effector pose along the whole path but the methods are different. They run ten times in the test environment (a) and the average planning results are depicted in Table 2.
Planner configurations are given in the Appendix as well.
The Cartesian Constrained EET planner outperforms the CBiRRT planner significantly in the planning scene (a). Reasons are:

(1) The Cartesian Constained EET planner considers the constraints already when it samples the end-effector poses, while the CBiRRT planner needs to project iteratively the sampled joint configurations on the constraint manifold with large computational burden. This complicated method results in much lower sampling efficiency and longer planning time.

Table 1. Planning result of the SNS-integrated EET Planner and the basic one in the test environment (b)

Result (Average value)	**Value**	
	SNS-integrated EET planner	Original EET planner
Planning time (ms)	790.2	1083.4
Number of samples	48.2	59.8
Satisfy rate of collision detection (%)	91.8 (2613.8/2847.8)	89.8 (3135.0/3493.0)
Success rate of extend operation (%)	92.0 (508.6/553.1)	88.8 (504.1/567.9)
Number of tree nodes	471.0	527.0
Path length in euclidean space (mm)	1032.9	1052.4
Path length in configuration space (rad)	9.8	10.1
Failure rate (%)	2.0	6.0

Table 2. Planning results of the Cartesian Constrained EET Planner and the CBiRRT planner in the test environment (a)

Result (Average value)	**Value**	
	Cartesian Constrained EET planner	CBiRRT planner
Planning time (ms)	897.9	5847.8
Number of samples	38.2	10370.2
Satisfy rate of collision detection (%)	97.1 (539.0/556.0)	83.2 (2828.4/3400.0)
Number of tree nodes	520.2	1459.0
Path length in euclidean space (mm)	820.6	5944.6
Path length in configuration space (rad)	8.6	21.9
Failure rate (%)	10.0	0.0

(2) The CBiRRT planner does not sample the joint configurations towards the goal configuration. Some samples have no contribution to finding an optimal shorter path solution but require large memory consumption, which leads to much longer paths and more tree nodes. In contrast, the EET planning is guided by the sphere tunnel directly towards the goal.
(3) Whether the constraints are rigorous or not has no influence on the performance of the Cartesian Constrained EET planner due to its sampling scheme, while the CBiRRT planner will have even worse performance with critical restrictions.

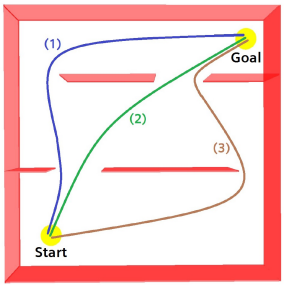


Fig. 3. Three homotopically different paths in test environment (c)

Parallelized EET Planner Unlike other extended EET planners presented above, the parallelized EET planner does not improve its capability of finding one better solution in a more efficient way, but shortens the duration for generating several paths. Therefore, comparing the quality of paths generated by the EET planner with and without parallelization does not make much sense because the quality of each path should be the same. But evaluation among the homotopically different paths is necessary to select an optimal one regarding some criteria. To figure out if the parallelization brings some benefits, only planning time is assessed.
Planner configuration in this experiment in the test scene (c) is given in the Appendix. Three alternative possible routes for the end-effector to go from the start to the goal position are depicted in Fig. 3.
Each planner solves the planning problem 50 times repeatedly and the average values are analysed. As a result, it takes the EET planner without parallelization 2739.5 milliseconds to execute all three alternative paths sequentially (one after another) in the labyrinth, while the parallelized EET planner needs 1240.5 milliseconds, which exhibits over 50% reduction of the planning time.
Furthermore, properties of the three different paths are given in Table 3. The planning time of the parallelized EET planner is almost equal to the longest planning time of a single path. Planning time of the EET planner without parallelization is almost equal to the sum of the planning time used for every path. Apparently, path 2 in Fig. 3 is the optimal path regarding the planning time and the path length. The benefits of applying parallelization in the EET planning are summarized here:

(1) Assuming that there are different homotopy classes of paths in the planning environment, the parallelized EET planner can generate several path solutions simultaneously and shorten the total planning time.
(2) The shortcoming in section 2 due to the incompleteness is compensated by looking for alternative solutions by using the parallelized EET planner, which enhances planning success rate.
(3) With several path solutions, some criteria such as path length in workspace can be considered to select the optimal path.

As a matter of fact, operations such as sharing parameters and assigning data memory for each process must be made additionally before the parallelization. Presence of different homotopy classes of paths in the environment is prerequisite for parallelization as well.

Table 3. Properties of the three alternative paths in the test environment (c)

Result (Average value)	**Value**		
	Path 1	Path 2	Path 3
Planning time (ms)	635.0	458.8	1186.9
Number of samples	76.4	44.1	147.8
Satisfy rate of collision detection (%)	88.7 (824.8/930.2)	89.1 (680.8/764.0)	79.0 (968.2/1225.6)
Number of tree nodes	261.4	203.6	303.6
Path length in euclidean space (mm)	950.1	748.7	977.4
Path length in configuration space (rad)	4.8	3.8	5.4

5. CONCLUSIONS

We present three variants of the Exploring/Exploiting Tree (EET) Motion Planner with three advanced features: a) consideration of the hard bounds of robot joint motion, b) the Cartesian constraints on the robot end-effector poses and, c) planning parallelization to find alternative solutions.

Integrating the SNS algorithm into the EET planner leads to combination of sampling-based motion planning and reactive control, which provides benefit of satisfying the hard joint motion limits. Taking advantage of the distinct sampling scheme in the EET planning, preservation of Cartesian constraints on the end-effector pose can be guaranteed efficiently. For mitigation of the incompleteness and for optimal planning the parallelization is introduced in the EET planner. Presence of different homotopy classes of paths is prerequisite.

According to the experimental results, our three variants of the EET planner outperform other motion planners with respect to several criteria in our test scenarios. In our future work, we will consider the performance of our planners in dynamically moving obstacles and the further extension to hierarchical stack of tasks Flacco et al. (2015).

ACKNOWLEDGMENT

This work was partly supported by the German Federal Ministry of Education and Research (BMBF) through the Hybr-iT project (grant no. 01IS16026A).

REFERENCES

Albu-Schäeffer, A., Haddadin, S., Ott, C., Stemmer, A., Wimböck, T., and Hirzinger, G. (2007). The dlr lightweight robot – design and control concepts for robots in human environments. *INDUSTRIAL ROBOT-AN INTERNATIONAL JOURNAL*, 34, 376–385.

Berenson, D., Srinivasa, S.S., Ferguson, D., Collet, A., and Kuffner, J.J. (2009). Manipulation planning with workspace goal regions. In *2009 IEEE International Conference on Robotics and Automation*, 618–624.

Berenson, D., Srinivasa, S.S., Ferguson, D., and Kuffner, J.J. (2009). Manipulation planning on constraint manifolds. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 625–632. IEEE.

Bhattacharya, S. (2011). Identification and representation of homotopy classes of trajectories for search-based path planning in 3d.

Bhattacharya, S., Kumar, V., and Likhachev, M. (2010). Search-based path planning with homotopy class constraints. In *Third Annual Symposium on Combinatorial Search*.

Brock, O. (2000). *Generating Robot Motion: The Integration of Planning and Execution*. Ph.D. thesis, Stanford, CA, USA. AAI9961867.

Brock, O. and Kavraki, L.E. (2000). Decomposition-based motion planning: Towards real-time planning for robots with many degrees of freedom. Technical report.

Brock, O. and Khatib, O. (2002). Elastic strips: A framework for motion generation in human environments. *The International Journal of Robotics Research*, 21(12), 1031–1052.

Devaurs, D., Siméon, T., and Cortés, J. (2011). Parallelizing rrt on distributed-memory architectures. In *Robotics and automation (ICRA), 2011 IEEE International Conference on*, 2261–2266. IEEE.

Flacco, F., De Luca, A., and Khatib, O. (2012). Motion control of redundant robots under joint constraints: Saturation in the null space. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 285–292. IEEE.

Flacco, F., De Luca, A., and Khatib, O. (2015). Control of redundant robots under hard joint constraints: Saturation in the null space. *IEEE Transactions on Robotics*, 31(3), 637–654.

Gammell, J.D., Srinivasa, S.S., and Barfoot, T.D. (2014). Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2997–3004.

Jacobs, S.A., Stradford, N., Rodriguez, C., Thomas, S., and Amato, N.M. (2013). A scalable distributed rrt for motion planning. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 5088–5095. IEEE.

Karaman, S. and Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104, 2.

LaValle, S.M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, U.K. Available at http://planning.cs.uiuc.edu/.

LaValle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning.

Marín, P., Hussein, A., Martín Gómez, D., and de la Escalera, A. (2018). Global and local path planning study in a ros-based research platform for autonomous vehicles. *Journal of Advanced Transportation*, 2018, 1–10.

Mesesan, G., Roa, M.A., Icer, E., and Althoff, M. (2018). Hierarchical path planner using workspace decomposition and parallel task-space rrts. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1–9.

Quinlan, S. (1994). *Real-Time Modification of Collision-Free Paths*. Ph.D. thesis, Stanford, CA, USA.

Rickert, M., Brock, O., and Knoll, A. (2008). Balancing exploration and exploitation in motion planning. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2812–2817. IEEE.

Rickert, M., Sieverling, A., and Brock, O. (2014). Balancing exploration and exploitation in sampling-based motion planning. *IEEE Transactions on Robotics*, 30(6), 1305–1317.

Siciliano, B. (1990). Kinematic control of redundant robot manipulators: A tutorial. *Journal of intelligent and robotic systems*, 3(3), 201–212.

Siciliano, B. and Khatib, O. (2016). *Springer handbook of robotics*. Springer.

Appendix A. CONFIGURATION OF THE MOTION PLANNERS

Table A.1. Configuration of the SNS-integrated EET Planner, the Cartesian Constrained EET Planner and the original EET Planner

	Parameters	**Value**
	Time step size (s)	0.05
	Desired task velocity v_{des} (mm/s)	50.0
	α	0.02
	β	0.30
	γ	0.33
	Tolerance to pose (mm)	5.0
	Clearance to obstacle (mm)	10.0
Maximum planning time (s)		120.0

Table A.2. Configurations of the CBiRRT planner

CBiRRT Planner		
	Parameters	**Value**
	ϵ	0.1
	Configuration step size	0.5
	Clearance to obstacle (mm)	10.0
Maximum planning time (s)		120.0

Table A.3. Configuration of the parallelized EET planner

Parallelized EET Planner		
	Parameters	**Value**
	Configuration step size (rad)	0.02
	Cartesian Product Distance	0.1,0.9
	α	0.01
	β	0.10
	γ	0.33
	Tolerance to pose (mm)	1.0
	Clearance to obstacle (mm)	10.0
	Parallelization	
	Thread number	3
Maximum planning time (s)		120.0