Robotics and Autonomous Systems 89 (2017) 123–135

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Autonomous navigation for reconfigurable snake-like robots in challenging, unknown environments

L. Pfotzer [*], S. Klemm, A. Roennau, J.M. Zöllner, R. Dillmann

Intelligent Systems and Production Engineering (ISPE), FZI Research Center for Information Technology, Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany

HIGHLIGHTS

- Fully integrated path and motion planning approach for modular reconfigurable robots.
- Planning system adapts itself to different robot configurations.
- Consideration of different capabilities for overcoming obstacles.
- Newly introduced Secondary Nearest Neighbor (SNN) space extends the RRT* algorithm.

ARTICLE INFO

Article history:
Available online 2 December 2016

Keywords:
Snake-Like robot
Autonomous navigation
Motion planning
Reconfiguration
Unstructured environment
Stair climbing

ABSTRACT

We present a navigation approach for reconfigurable snake-like robots to autonomously overcome unknown and challenging obstacles like stairs or large steps. To calculate convenient motions we use a planning algorithm that takes different optimization criteria like distance, time or energy into account. In addition, the current robot configuration, i.e. the combination of robot modules that make up the complete system, is considered while planning, whereby the planning algorithm can determine whether an obstacle can be overcome or not. To handle constraints like inholonomity of the robot we use a purely geometric planner together with an extension, the *Secondary Nearest Neighbor Space*, to increase the efficiency of the planning algorithm. Combining multi-stage navigation and motion planning with a follow-the-leader control method keeps the planning time unaffected from a changing amount of robot modules. The developed and implemented methods are presented and three different scenarios have been chosen for evaluation on a real robot: the reconfigurable snake-like robot KAIRO 3. The results show that our approach is very well-suited to distinguish different robot configurations and enables snake-like robots to overcome previously unknown obstacles fully autonomously.

1. Introduction

Autonomous navigation is a very important skill of modern mobile robots. The main goal of conventional navigation and motion planning approaches is, using *obstacle avoidance* to generate collision free paths. With the increasing amount of various biological inspired robots, for instance snake-like robots, new locomotion opportunities arise. Transeth et al. [1] introduced the term *obstacle-aided locomotion* to highlight the possibility of snake-like robots *integrating* obstacles into their locomotion. Since biologically inspired robots are able to step, move or climb over obstacles, new and specialized control and navigation strategies need to be developed, which take advantage of those extended locomotion capabilities. Furthermore, modular and reconfigurable robots are able to change their kinematics and configuration.[1] This requires more flexibility and adaptability of the overall robot control and planning system.

To realize a holistic navigation system, we choose a combined path planning and locomotion control based approach instead of a pure control based method. This also enables over-the-horizon navigation, i.e. navigation in only partially observable areas of the environment with a receding sensor horizon, and taking long distances and large areas of complex environments into account without getting stuck in local minima—one of the drawbacks of classic control based approaches. Furthermore, specialized and optimized intermediate paths, fitting to the configuration and capabilities of robot, are generated for short distances in front of the

[*] Corresponding author.
E-mail addresses: pfotzer@fzi.de (L. Pfotzer), klemm@fzi.de (S. Klemm), roennau@fzi.de (A. Roennau), zoellner@fzi.de (J.M. Zöllner), dillmann@fzi.de (R. Dillmann).

[1] In this context, a robot configuration is denoted as the assembly and connection of compatible hardware modules to form a robot system. Reconfiguration means changing the structure and connection of these hardware modules.

http://dx.doi.org/10.1016/j.robot.2016.11.010
0921-8890/© 2016 Elsevier B.V. All rights reserved.

robot which are then executed by an adaptive follow-the-leader control approach.

2. Related work

In mobile robot navigation a large range of motion planning algorithms are utilized to generate safe paths for a steadily growing number of applications and robotic systems. The variety of planning approaches ranges from local methods like potential fields that repel robots from obstacles [2] to graph based planning using motion primitives [3,4] that abstact away the complexity of the nonholonomic nature of many robots. For planning in high-dimensional state spaces sampling based algorithms are found to perform efficiently, e.g. consider the Rapidly-exploring Random Tree (RRT) introduced by LaValle [5] or the RRT-Connect by Kuffner [6]. These algorithms find *feasible* paths, but make no guarantees about the quality of the planning results. In [7] and [8] Karaman et al. introduce the *asymptotically optimal* motion planning algorithm RRT* and give an in-depth overview of single and multi query motion planning algorithms and the underlying sampling theory. A general yet detailed overview about the broad field of motion planning is found in fundamental books by Latombe et al. [9], LaValle [10], Siciliano et al. [11] and Coenen et al. [12].

The previously mentioned work is applied to a large variety of path planning problems for mobile, wheeled robots. In our work we focus on the challenging problem of generating feasible optimal paths for a robot with a snake-like kinematic and a varying number of modules that build up the robot's central body. For such a bio-inspired inspection robot, that should be applicable in unstructured terrain, a number of opposing requirements arise. Thus, the complex snake-like robot must be modeled in a way that provides efficient motion planning. In this work we propose a multi-layered, biologically inspired approach to create whole body motions suitable for avoiding and overcoming obstacles.

A basic form for snake-like robot locomotion is constituted by fixed motion patterns, which are implemented in different forms by Tsakiris et al. [13] and Bayraktaroglu et al. [14]. More sophisticated approaches use so called central pattern generators [15,16] or alternatively backbone curves [17] which mime biological motions. Hirose et al. [18] investigated motion generation for snake-like robots using a 3-D active cord mechanism (ACM) to implement different gaits. Adapting locomotion patterns to the underlying terrain, Chirikjian et al. [19] developed mathematical models for inextensible and extensible mechanisms using a given path. They do not consider different robot configurations when generating the path.

Besides the pure snake-like locomotion principles there are some robotic systems that follow a hybrid approach by additionally equipping the robot with wheels, e.g. consider OmniThread [20], Wheeeler [21], KOHGA [22] and the robot KAIRO 3 [23] which is used for evaluating the navigation framework proposed in this work. Basic motion of wheeled snake-like robots can be achieved by the *follow-the-leader* approach as shown by Scholl et al. [24] and Yamada et al. [25]. A further method is called *n-trailer* and illustrated by Granosik et al. [26], Murugendran et al. [27] and Altafini et al. [28].

In the context of *obstacle-aided locomotion*, the *scaffold-based locomotion* is introduced by Kano et al. [29]. This approach realizes a decentralized control scheme using obstacles to improve the movement of the snake-like robot HAUBOT III. EARLI (Extended Asymmetrical Reverse Lateral Inhibition) [30] is a torque based control law for obstacle aided locomotion of snake-like robots. By applying additional torque, the contact between the robot body and obstacles is exploited to propel the robot forward. A physics simulation is used for the evaluation of EARLI. Liljeback et al. [31] proposed a controller, which is based on the principle of rotating contacted links to increase the propulsive force for obstacle-aided snake locomotion. The implemented controller uses the *Leader–Follower Scheme* to realize a propagation of the forward propulsion through the body of the snake robot.

While conventional wheeled robots require a navigation approach that provides safe path generation and execution to avoid obstacles, biologically inspired robots like KAIRO 3 (see Section 3) and LAURON V [32] are technically able of also climb over obstacles. This capability however arises further complexity for the motion planning algorithm as more complex motions may have to be taken into account and special planning strategies have to be considered. Another challenge is the ability of a robot to reconfigure and therefore have a varying kinematics. Currently only few navigation approaches are capable of handling such kinematic changes at runtime.

The primary goal of the proposed navigation system is increasing the autonomy and flexibility of mobile snake-like robots to make them more independent from human operators. Thus, we introduce a motion planning approach suitable for generating feasible and optimal motions for robots with many degrees of freedom (DOF) that can overcome obstacles in unstructured terrain. The resulting motions are viable for multiple robot configurations[2] and meet adjustable optimality criteria such as distance to drive and time consumption of the generated trajectory. Using a planning approach we aim for a predictable behavior of the robot, that does not get stuck in local minima, either while planning coarse routes towards points of interest in the environment or while planning motions to traverse the robot's near surrounding. With our multi-layered navigation approach we endeavor to reduce the complexity of the nonholonomic system and provide trajectories that are suitable for direct robot control.

The navigation system is capable of planning complex 6-D robot motions for the central body in the special euclidean group SE 3 (position and orientation in 3-D ambient space). We show that the robot is capable of overcoming obstacles autonomously while being able to handle multiple robot configurations and provide optimal locomotion taking into account the navigation capabilities of the robot. Our method uses a multi-layered navigation model with a three stepped motion planning process based on an extended RRT* algorithm and a multi-resolution environment model providing detailed terrain information. The snake-like robot motion is then executed by a modified follow-the-leader control approach using 6-D trajectories with additional joint angles.

3. The modular snake-like robot KAIRO 3

KAIRO 3 (Karlsruhe's Autonomous Inspection RObot 3), shown in Fig. 1, is a biologically inspired wheel-driven snake-like robot developed at FZI Research Center for Information Technology. With its modular design, consisting of different types of compoundable modules, KAIRO 3 is also classified as reconfigurable system. From a kinematic and geometric point of view, two kinds of modules can be distinguished: *drive* and *joint* modules, which are both visualized in Fig. 2.

Each drive module consists of two powered wheels and freight space for integrating custom payload like application specific sensors, embedded computation systems and batteries. The drive modules are alternately connected with joint modules, whereby the minimal configuration requires at least two drive and one joint module. The joint modules consist of three actively driven axes as shown in the sketch in Fig. 3. Every axis is actively driven by a motor and a harmonic drive gear to be strong enough for lifting up multiple modules as depicted on the left side of Fig. 1. This configuration of KAIRO 3 contains of six drive and five joint

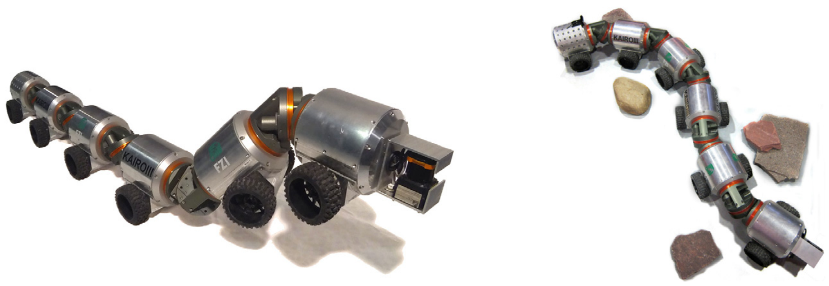


Fig. 1. The modular snake-like inspection robot KAIRO 3 [23] is able to lift up several body segments, e.g. the head as shown on the left. The right figure shows KAIRO 3's ability to move through narrow places.

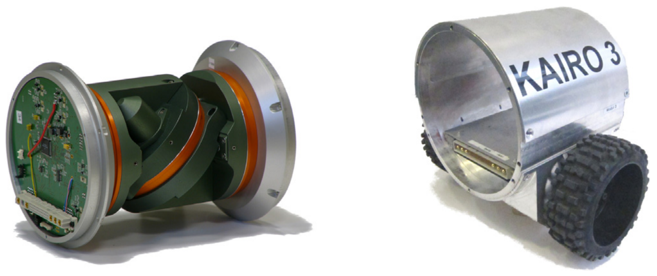


Fig. 2. A joint module of KAIRO 3 is shown on the left and a drive module with two individual controlled wheels is depicted on the right.

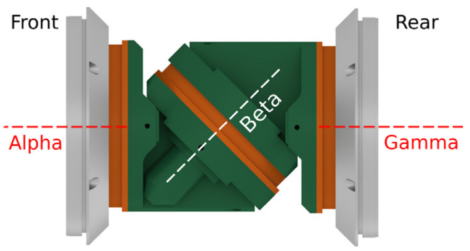


Fig. 3. Sketch of a joint module with three axes: Alpha (α), Beta (β) and Gamma (γ). The α and γ axis are aligned along the drive direction while the β axes is tilted by 45°.

modules with a total of 27 degrees of freedom (DOF), a length of about 1.8 m and a weight of about 47 kg including batteries.

KAIRO 3 is capable of moving around sharp corners in tight spaces as the joint modules can bend up to 90° along the vertical and lateral axis. Furthermore, it is able to pass through small openings and pipes with a minimum diameter of 25 cm due to its slim snake-like design. Thus, KAIRO 3 is applicable to many different purposes e.g. search and rescue missions, inspection and maintenance of structures with narrow entries or environments which are dangerous or hazardous to humans.

3.1. Adaptive control software

The *Adaptive Control* architecture of KAIRO 3 is implemented in the robotic framework MCA2 (Modular Controller Architecture) [33]. This hierarchical control software is able to adapt to the actual amount of hardware modules. Fig. 9 illustrates the connection between the *Adaptive Control*, consisting of two layers, and the navigation planning system.

The decentralized *Base Control* is a mixture of hardware and software components located in every module of the robot. Low level control algorithms run on a customized circuit board named UCoM (Universal Controller Module) [34], consisting of a digital signal processor (DSP) and a field programmable gate array (FPGA). In addition to driving up to three motors with pulse width modulation (PWM) signals, it reads motor encoder values, optical rotary encoders and measures motor currents. The communication between all UCoMs and the centralized control software is realized by a Controller Area Network (CAN) Bus.

On top of the *Base Control*, the kinematics calculation of KAIRO 3 as well as basic maneuvers like *Normal Drive* and *Inspection* are realized within the *Motion Control*, which uses a virtual rail.[3] Furthermore, the two biologically inspired locomotion methods *Caterpillar* and *Inchworm* [35] allow KAIRO 3 to move in uneven terrain without using its wheels.

3.2. Virtual rail motion control

The basic virtual rail [24] for robot motion control is build by several way points consisting of poses in SE 2, which are interconnected by linear line segments. Fig. 4 shows an example consisting of several way points.

Fig. 5 shows a robot consisting of three drive modules and two joint modules placed on top of a straight rail. All center points of the joint modules (kinks) follow this rail. Virtual joints (*kink #0* and *kink #3*) are inserted at the front and at the end of the robot to enable steering. At any time the kinks need to stay on the rail while the drive module center points may leave the rail in corners.

The forward movement of KAIRO 3 on the straight rail is shown in Fig. 6. The Cartesian positions of the drive modules are calculated from the interpolated kink positions between the neighboring rail way points. Starting from the pinned kink position, the inverse kinematics is used to calculate the joint angles for the joint modules and the drive module wheel velocities.

In case of curvatures, the rail will be bended by moving the way points in front of the robot towards the desired direction. Enlarging the rail is achieved by adding new path segments or inserting new way points in front of the *last rail base*. The *first rail base* and *last rail base* are the start points of a ray growing into the infinity. This means, the robot will keep its current orientation and speed until new way points are inserted in front of the *last rail base* (see Fig. 7).

In Section 4.4.1 an extension of the original virtual rail, utilizing 6-D way points with additional joint angles, will be introduced.

3.3. Reconfiguration

Due to the modular design of KAIRO 3, several different configurations can be realized. A single robot configuration with six drive and five joint modules is shown in Fig. 1. As depicted in Fig. 8, it is possible to form multi-robot systems, either with two

[2] In the case of KAIRO 3, multiple robot configurations are realized by varying the number of robot body modules.

[3] The virtual rail is a specialized and customized implementation of the follow-the-leader approach [24].

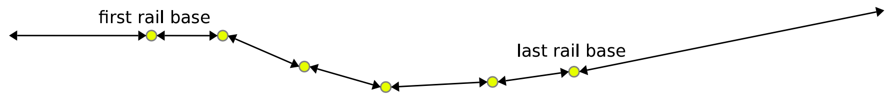


Fig. 4. Basic definition of a virtual rail with the first and last way points (*rail base*) as well as intermediate points describing a path the robot needs to follow.

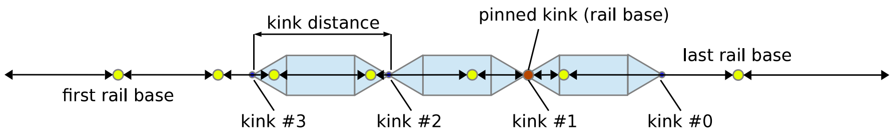


Fig. 5. Definition of a straight virtual rail with KAIRO 3 placed on top of it. The *kink distance* describes the length between the center points (kink) of two neighboring joint modules. One joint center point is marked as *pinned*, which indicates the default position.

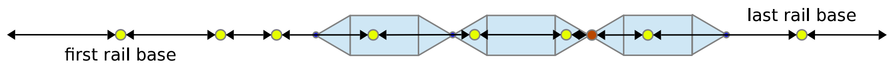


Fig. 6. Forward movement of the robot along the straight virtual rail.

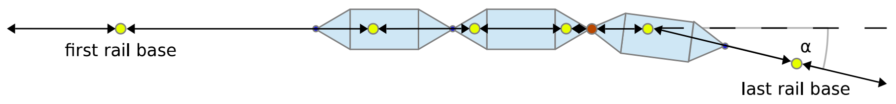


Fig. 7. Forward movement of the robot along a curved virtual rail. The last (*last rail base*) and second last way point are bent by angle α.

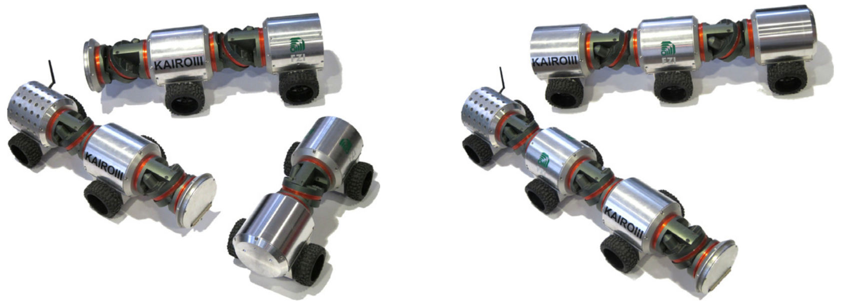


Fig. 8. The left configuration comprises three single robots with at least three modules, which is the most compact configuration with the minimal amount of modules. On the right image two robots with five as well as six modules form another configuration.

or three standalone and fully operational robots. To enable self-reconfiguration of KAIRO 3, we developed a mechanical interface as well as algorithms and methods to automatically connect and disconnect single modules without human interaction [23].

4. Configuration dependent navigation and motion planning

Considering modular and reconfigurable robots, the variation of their configuration has a large influence on navigation and motion planning. For example a long snake-like robot may overcome obstacles of a certain height, while robots with different kinematics and shape, respectively a shorter body length, need to find a way around the same obstacles. Starting from this point of view, we thought about a general navigation framework, which is based on a hierarchical path planning approach. The complexity of the requirements for the framework should be diminished by using a multi-layer motion planning system, consisting of several planning and execution stages, that should also be able to take multiple different robot configurations into account. Thus, we aim to plan paths for reconfigurable robots, which are able to overcome obstacles instead of avoiding them. The overall structure of this approach, is shown in Fig. 9.

In order to realize such a planning system, including the high-level planning system but also the motion control part of the robot (*Adaptive Control*), several existing frameworks and tools, have been considered. Especially, for the real time based *Adaptive Control* software [23,36], which directly communicates with the real robot, we chose the hierarchical Modular Controller Architecture (MCA2[4]). Due to its modular software structure, MCA2 is scalable and independent of the amount of robot modules. It integrates a decentralized Software and Hardware solution using a CAN bus to enable real time data exchange and control. This also enables the synchronization of all joint angle movements on the low-level motion control system (see also Section 3.1).

For the high-level planning stages, environment modeling and the user interface, we decided to base our system on the Robot Operating System (ROS[5]). Due to the *publish and subscribe* architecture, this middleware is very well-suited for transporting data structures between different software components. For example the environment map data in Fig. 9 are transmitted from the *Environment Modeling* to each planning component located at different levels in the software. Furthermore, ROS is capable of storing and transporting very large data, like 3-D point clouds.

Additionally, the Planetary Exploration and Navigation (*PlexNav*) framework [37] is utilized. *PlexNav* is based on ROS and

[4] Modular Controller Architecture (MCA2): www.mca2.org.
[5] Robot Operating System (ROS): www.ros.org.

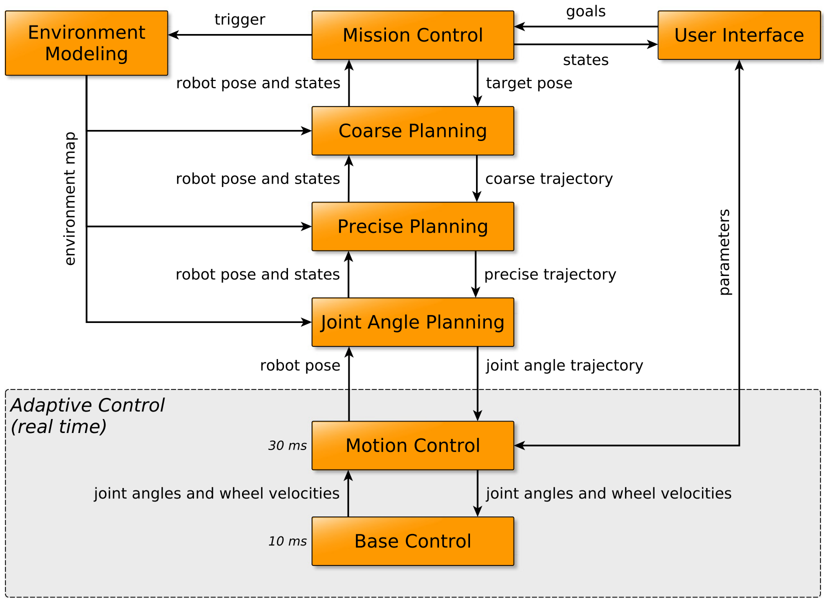


Fig. 9. System architecture of the integrated navigation and motion planning approach.

already consists of configurable planning algorithms, 3-D environment modeling and mapping as well as a basic mission control system.

The connection between the two worlds of MCA2 and ROS is managed by a MCA2 to ROS bridge, where a MCA2 module can also act like a ROS node and vice versa.

4.1. Mission Control and User Interface

The coordination of the planning system is realized by the *Mission Control* shown in the center top of Fig. 9. In particular, the procedure of a mission is supervised and controlled to achieve the goals and constraints given through the *User Interface* . This is the only stage where a direct interaction of a human operator with the entire system is necessary. An operator specifies either a single goal or a list of navigation goals, which encode different target poses.[6] Then, the *Mission Control* triggers the *Environment Modeling* to collect information about the environment and to construct a map. Improving the quality and coverage of the mapping, also with varying arrangement of sensors, the *Mission Control* has been extended by special robot dependent movements denoted as *Mapping Maneuver*. Such a *Mapping Maneuver*, which is particularly adapted to the applied robot kinematics, is integrated into the *Mission Control* by a plug-in mechanism and can be utilized as special planning goal. External events like the detection of a step or gap as well as temporal or spatial repeated impulse can be used as a trigger for the *Mapping Maneuver*. Since snake-like robots normally consist of a small diameter, sensors are placed in a very low position over the floor and suffer from occlusions and hidden objects. With this technique, a *Mapping Maneuver* for snake-like robots was implemented to vertically lift up a sensor, mounted on the robot's head, to get a better overview of the terrain and thus more accurate maps and cover a larger area with fewer sensor recordings. Especially the drivable ground plane can be detected more precisely when using a higher sensor position.

[6] A target pose consists of six dimensions (x, y, z, *roll*, *pitch*, *yaw*) and describes the goal position and orientation in the Special Euclidean group SE 3 for the first robot module.

4.2. Environment Modeling

The *Environment Modeling* combines perception, mapping and localization by utilizing the *PlexMap* [37] environment map representation as integral part of the PlexNav Framework. With this generalized 2.5-D map, multiple 3-D point clouds can be merged to a single map. Thus, the *PlexMap* can integrate data from a large variety of different 3-D sensors into a uniform map representation. In this work, the high resolution rotating laser scanner KaRoLa [38] was used to capture 3-D point clouds because it fits very well to snake-like robots due to its compact size.

Adding a new point cloud to the map is achieved by a modified iterative closest point (ICP) based algorithm registering the new with the previously recorded point cloud. The robot odometry is used to provide an initial pose estimate to the ICP algorithm. Additionally to the registration, a loop-closing between spatial connected point clouds is executed to reduce the uncertainty of the robot's localization.

A *PlexMap* is a quad tree data structure that consists of *cells* with different sizes dividing the three-dimensional Cartesian space into small rectangular sections. The cell sizes are dynamically and regionally adapted to the amount and distribution of the measured points. Next to the height values, the *PlexMap* also stores inclination and terrain roughness for each cell. Both values are derived from a plane estimation deduced from a principle component analysis of the sub point cloud delimited by the cell bounds. The normal vector onto the plane can be used to calculate the inclination of the terrain at this cell, and the variance of the sub point cloud from the estimated plane is interpreted as terrain roughness. Thus, the *Environment Modeling* provides a precise mapping and localization system which is used by the *Trajectory Planning* component.

4.3. Trajectory Planning

In general, methods for path and motion planning create different kinds of trajectories which are executed by the robot control system. Especially snake-like robots with a lot of DOF,[7] require fast planning methods like Rapidly-Exploring Random Trees [39] or

[7] Remember KAIRO 3 has 27 DOF in joints plus six for the pose of the base point.

Randomized Kinodynamic Planning [40] to cope with the high dimensional planning state space efficiently. With these approaches, full mathematical models of the robot kinematics including motion constraints are often used to consider all joints and poses during the planning process. Such a complete model results in a heavy computational load, especially when also inverse kinematics has to be calculated. Additionally, considering robots with variable kinematic chain length, the runtime of direct and inverse kinematics scales badly with increasing kinematic chain length as more robot modules and joints cause a higher dimensionality of the state space used for planning. This may lead to a bloat of the search space according to the curse of dimensionality [41]. With the usage of reconfigurable robots, variations of the robot configuration itself get more significant and thus changes in the kinematics model and motion constraints are required on-line.

Due to the drawbacks of using a full kinematic model for path planning, we propose a method which uses a reduced search space and multiple planning stages. As illustrated in Fig. 9, the path planning process is divided into the three steps *Coarse Planning*, *Precise Planning* and *Joint Angle Planning*. The search space is kept constant for all planning stages, also with a variable amount of robot modules, by planning 6-D poses only for the first robot module. The *Joint Angle Planning* step additionally takes care of the joint angles which have been reduced to consider only neighboring modules. These simplifications are possible for modular snake-like robots due to a combination of the planning process with the *follow-the-leader* motion control approach (Section 3.1). In this way the combined navigation and motion planning system is independent from a changing amount of modules while being able to handle different configurations of snake-like robots.

4.3.1. *Coarse Planning and Precise Planning*

The *Coarse Planning* determines a path of SE 3 poses for the robot origin[8] from the current robot pose to a target pose given by the *Mission Control* . Poses outside of the already created environment map and sensor horizon of the robot are allowed. This kind of planning is also denoted as over-the-horizon planning.

In a second step, the *Precise Planning* refines the next subsection of the coarse plan which is inside an already known map area. The resulting path also consists of 6-D poses for the robot origin as way points which correspond to the best path the robot could take regarding the given environment information and current robot configuration. At this stage the path does not contain any information about how the robot can move along the path, over obstacles or gaps.

The two planning steps *Coarse Planning* and *Precise Planning* are implemented utilizing an extended version (see Section 5) of the RRT* algorithm [42]. The cost function, which is specialized for modular snake-like robots, is explained in Section 5.1. This cost function adjusts to the amount of robot modules and the robot configuration and thus influences the resulting path. For example a short snake-like robot will get very high (up to infinite) costs for an obstacle which is very high or even too high for climbing in comparison to a longer robot. Further improvements of the planning results, especially for nonholonomic motions, are achieved by the *Secondary Nearest Neighbor Space* extension for RRT*, which is introduced in Section 5.2. The main difference between *Coarse Planning* and *Precise Planning* are modified cost function parameters and a reduced step size for the *Precise Planning*.

[8] In case of KAIRO 3, the robot origin is located in the center point of the first drive module.

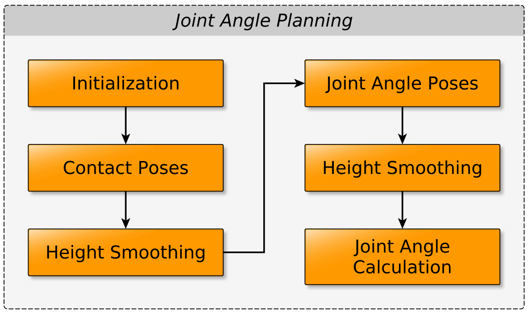


Fig. 10. Joint angle planning process which consists of six phases.

4.3.2. *Joint Angle Planning*

In the third planning stage, *Joint Angle Planning*, a subsection of the precise path is taken to generate a smooth path adapted to the terrain respectively surface of obstacles regarding the geometry and kinematics of the robot. Since this step depends on the deployed robot modules, it has to be adjusted for different types of robots. In the following we are considering a wheel-driven modular snake-like robot with drive and joint modules similar to KAIRO 3. The joint angle trajectory needs to provide poses for the joint modules' center points according to the virtual rail. However, the coarse and precise planning steps generate trajectories for the robot origin which is located in the center of the first drive module. Thus, the way points of the precise plan have to be transformed and also new trajectory poses have to be added in cases where joint angles need to be changed. These additional poses are mainly required in path sections next to obstacles like steps. Furthermore, the explicit specification of joint angles also realizes a synchronization between the neighboring joint modules.

The process of joint angle planning, as shown in Fig. 10, starts with the *Initialization* phase which determines the currently relevant section of the *PlexMap* and the virtual rail part representing the current robot state. Based on the actual robot pose the plausibility of the received path (from *Precise Planning*) is examined by iteratively testing the *yaw* angle changes and the absolute *pitch* angles.

Starting from the current robot pose, new states are created towards the target pose, using a fixed step width. For each of these states a *Contact Pose* is created to project the robot to the ground at given x, y and *yaw* coordinates. The resulting height z for the first drive module is determined from the average heights of both wheels provided by the *PlexMap* . Furthermore, the *roll* angle Φ is derived from the height difference of the wheel contact points h_l, h_r and the distance between the wheels d:

$$\Phi = \arcsin\left(\frac{h_r - h_l}{d}\right). \qquad (1)$$

Within the *Height Smoothing* phase, the generated poses are filtered and smoothed to compensate map inaccuracies and to allow overcoming obstacles:

1. Moving Median Filter: Removes rough outliers.
2. Relative Threshold Filter: Suppresses sensor noise.
3. Linear Interpolation: Discrete height steps and connection of steps, if the connection line does not differ much from the ground (for ramp following).
4. Moving Mean Filter: Creates fluent transitions between interpolated sections.

Before calculating joint angles, the *Contact Poses* have to be transformed into *Joint Poses*, which are used by the virtual rail. This translation is done by a fixed step width. Especially, large *pitch* or *yaw* angle changes of the path may result in joint poses located behind the previous pose in direction of movement. Due to verification of maximum angle changes between two following states, states which exceed the maximum angle threshold are filtered out.

The *Joint Poses* are again smoothed with the above height smoothing functions before enriching the 6-D poses by joint angles α, β and γ.

Required *Joint Angles* for the first joint module are iteratively calculated using the inverse kinematics. To cope with constraints like electrical wiring inside the joint modules, that limit the amount of usable joint values, α and γ joint angles are turned back towards their zero position if they exceed 270° and the β joint is inverted, leading to a relaxation of the robot configuration away from joint interval limits. Additionally, α and γ joint angles are linearly interpolated across multiple states to prevent large changes particularly in sharp corners. Finally, the *Joint Poses* are smoothed using the *roll* angles calculated with Eq. (1) and the 6-D *Joint Poses* combined with the *Joint Angles* are transferred to the *Motion Control* .

4.4. Motion Control and Base Control

The *Motion Control* applies the 6-D *Joint Module Poses* and *Joint Angle* target values to the extended virtual rail (see Section 4.4.1). While the robot is moving with a given speed along the rail, the inverse kinematics calculates the intermediate joint angles for all robot modules at the same time. For automatically adjusting the drive speed to the joint angle movement, an alternating speed controller has been integrated, switching between two controllers for accelerating and braking.

4.4.1. Virtual rail extension

The original virtual rail of KAIRO 3, shown in Section 3.1, only consists of 3-D poses [36]. To allow also vertical movements, the virtual rail has been extended to 6-D poses for each way point of the rail. Additionally the joint angles α, β and γ are stored for every pose to rotate redundant joints in advance which saves time and also synchronizes the joint modules as illustrated in Fig. 11.

With the robot position between two rail points, the joint angles are interpolated using Eq. (2), where $\vec{\psi}$ denotes the joint angles and $\vec{p}$ the pose. The indexes *prev* and *next* address the previous and next rail way point and *curr* refers to the current way point.

$$\vec{\psi} = \vec{\psi}_{prev} + \frac{|\vec{p}_{curr} - \vec{p}_{prev}|}{|\vec{p}_{next} - \vec{p}_{prev}|} \cdot \left(\vec{\psi}_{next} - \vec{\psi}_{prev}\right). \tag{2}$$

5. Extended RRT* algorithm

As mentioned above, we use an extended version of the RRT* algorithm in multiple planning stages. The RRT* algorithm is a sampling based planning approach that has been shown to generate asymptotically optimal solutions. With increasing amount of given planning time, found solutions will converge towards a theoretically optimal plan. The optimization is done considering a cost function that takes into account application or robot specific preferences and penalties. Besides the *soft* optimization criteria there exist *hard* constraints that prevent the planning algorithm to generate invalid motions, e.g. such that would lead into an obstacle or violate the operational joint interval regarding the robot's kinematics. In the following sections we explain the adaptations made to the cost function compared to the PlexNav framework and introduce the *Secondary Nearest Neighbor Space* used to cope with the robot's inholonomity.

5.1. Cost function

In order to meet the nonholonomic properties of wheel-driven snake-like robots, the cost function proposed in PlexNav [37] has been adjusted and extended by additional constraints. Costs for reaching pose s_2 from pose s_1 are calculated with Eq. (3) consisting of three mathematical terms:

$$\begin{aligned} cost_{s_1,s_2} &= \textsc{Dist}(s_1,s_2) \cdot \frac{1}{N}\sum_{i=0}^{N} C(q_i) \\ &+ \epsilon \cdot \frac{(\textsc{Yaw}(s_1,s_2)+1)^2 - 1}{\textsc{Dist}(s_1,s_2)\cdot \pi} \\ &+ \zeta \cdot \textsc{OrientDiff}(s_2, s_t), \end{aligned} \tag{3}$$

where Dist represents the Euclidean distance, Yaw returns the yaw angle difference and OrientDiff sum up the 6-D orientation change between two poses.

The first cost term determines the average cost for all N intermediate states q_i[9] between s_1 and s_2, factorized with the Euclidean distance of both states. Each $C(q_i)$ represents the maximum of the weighted costs for the intermediate state q_i considering the following *PlexMap* parameters (see Section 4.2):

- height difference between the wheel ground contact points of two neighboring drive modules[10]
- terrain roughness value stored in the map cell at sampling state q_i
- cell size at sampling state q_i depending on the amount of scan points.

Inclination costs from the map are not directly included at this point due to the ability of snake-like robots to climb up vertical steps with a certain height. This maximum vertical height difference (climbing capabilities of the robot according to the current robot configuration) is already considered in first cost term. Additional to this main cost term, a second and third cost term have been introduced to take care of the nonholonomic properties[11] of wheel-driven snake-like robots.

The second term calculates the *yaw* angle difference between two following states depending on their Euclidean distance and normalized by π. This ensures the limitation of the possible curvature, regarding the maximum steering angle of robot, by an increase of the overall costs. The weight ϵ is used to control the influence of this cost term.

The third term aims to push robot towards the next intermediate target pose s_t and particularly prevents turning in place. For this purpose, the 6-D orientation difference between state s_2 and the intermediate target state s_t is used. The weight ζ is used to control the influence of this cost term.

5.2. Secondary nearest neighbor space

The extension and modification of the RRT* cost function, especially the constraints for nonholonomic movement, discards many of the randomly selected samples which significantly increases the planning time. This arises mainly during the nearest neighbor search between newly drawn samples and the existing tree, when

[9] Intermediate states q_i are interpolated with a parameterizable step factor between the two sampled poses s_1 and s_2, to get a more accurate coverage of the environment map.

[10] The maximum height difference which a robot is able to overcome is determined from the current robot configuration (amount and dimension of the central body modules).

[11] The property of a robot or vehicle which is not able to turn in place but requiring a forward or backward movement to turn with a thresholded steering angle is denoted as nonholonomic.

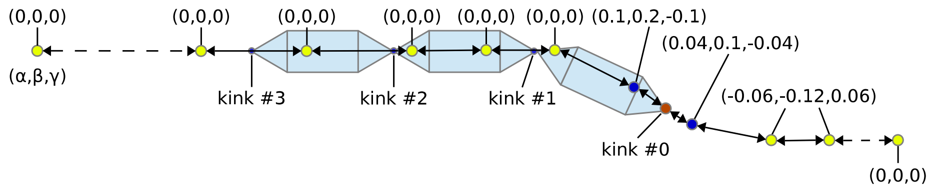


Fig. 11. Robot with five modules is moving along the extended virtual rail. Yellow nodes denote 6-D rail way points with assigned joint angles α, β and γ. The first virtual joint module (kink #0) is shown in red and lies in front of the first drive module. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

trying to find an appropriate connection. This step is a crucial part during tree generation for all RRT-like planning approaches and a variety of different effects may occur depending on the different possibilities of tree wiring, e.g. compare the non-optimality of the vanilla RRT algorithm and the slight modification in the tree wiring rule in RRT*, that forms a cornerstone for finding *asymptotically optimal* solutions. We found that the default Euclidean distance nearest neighbor search leads to violation of the minimum steering angle (*yaw* angle) constraint. This leads to a bad planning performance and unnecessarily lengthens the required planning time. To overcome this drawback we introduce the *Secondary Nearest Neighbor Space* (*SNN* space), that facilitates planning approaches to efficiently handle inholonomity constraints without having to use kinodynamic approaches. This is of special interest as purely geometric approaches tend to be computationally more efficient: the planning problem can be reduced from the kinodynamic (i.e. geometric and control) domain to a purely geometric state space.

The *SNN* space is comparable to a projection of the nearest neighbor connection for newly generated samples into direction of the root node of the RRT* tree. This projection results in connections respectively edges with angles mostly larger than the minimum steering angle.

Realizing the *SNN* approach, the RRT* tree is stored in two separated but related spaces: The primary space contains all nodes[12] and edges of the classical RRT* tree as shown in Fig. 12a.

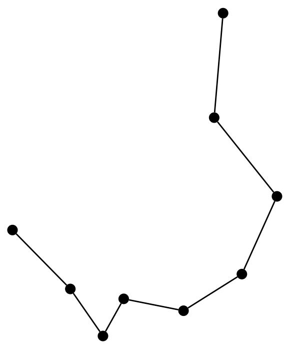
Next to the primary space a secondary space is added, which contains exactly one node for each node in the primary space. Secondary nodes are created by translating the primary node with a parameterizable distance δ. To find suitable values for δ, we have to consider that new samples are added with a maximum orientation change ϕ regarding their predecessor node. ϕ strongly depends on the ratio of the maximum RRT step width s and the selected translation δ for the *SNN* . Bounding the orientation change works for both *Pitch* and *Yaw* angles and will be smaller than 360°, if $\delta \geqslant s$:

$$\phi = arcsin\left(\frac{s}{\delta}\right). \tag{4}$$

The previously mentioned translation δ takes place in the same direction than the edge between the primary node and its predecessor in primary space. Edges in the secondary space (dashed lines) indicate the connection between a secondary node and its corresponding primary node. With these connections, it is possible to access nodes of the respectively other space. Adding the *SNN* space to Fig. 12a results in the extended RRT* tree illustrated in Fig. 12b.

To integrate the *SNN* data structure into the RRT* algorithm, the methods for inserting new nodes and tree reconnection had to be adjusted. Fig. 13a shows the insertion of a new randomly generated sample (red node). First of all, the nearest neighbor search takes place in the secondary space (indicated by the red line). With the relation between primary and secondary space, the corresponding node in the primary space is selected as predecessor of the new

[12] Each RRT* sample corresponds to a node in the RRT* tree and contains a 6-D pose representing the robot origin.


(a) Primary RRT* tree structure.

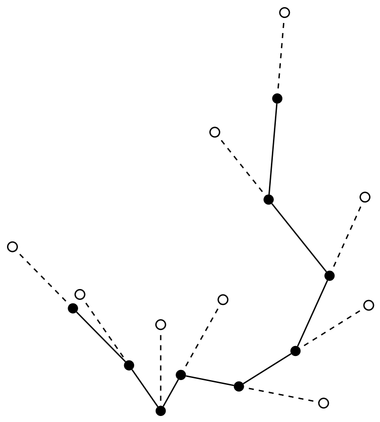
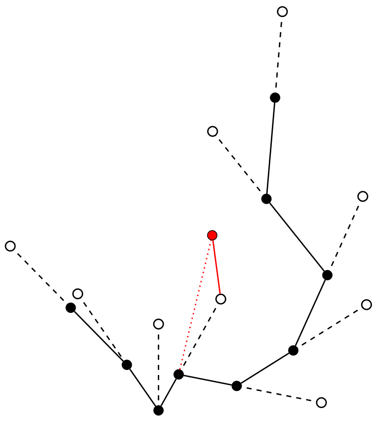

(b) RRT* with *SNN* extension.

Fig. 12. Extension of the RRT* algorithm by the *Secondary Nearest Neighbor Space*. Black filled nodes and lines belong to the primary space while correspondences of the secondary space are denoted as white filled nodes and dashed lines.

sample (indicated by the dashed red line). The new node is then added to the primary space, as shown in Fig. 13b, by translation from the predecessor node in direction to the sample. Finally, the corresponding node is added to the *SNN* space.

The process of reconnection (rewiring of the tree according to possible cheaper paths along the tree structure, see [7]) uses the *SNN* space to find shorter paths. For each node n_s in the *SNN* space, the nearest neighbor n_{p1} in the primary space is searched. A reconnection has to be done if the found nearest neighbor n_{p1} is not directly connected with the corresponding primary node n_{p2} of n_s in primary space. Fig. 14a shows an example where this rule is broken. The reconnection removes the connection in the primary space between the nearest neighbor n_{p1} and its predecessor in the RRT* tree. Afterwards the nearest neighbor n_{p1} is connected to the corresponding primary node n_{p2} of n_s in primary space. To ensure consistency, secondary nodes related to the modified nodes are updated regarding the new tree structure. The resulting RRT* tree with *SNN* space is show in Fig. 14b.


(a) The nearest neighbor search takes place in the *SNN* space.

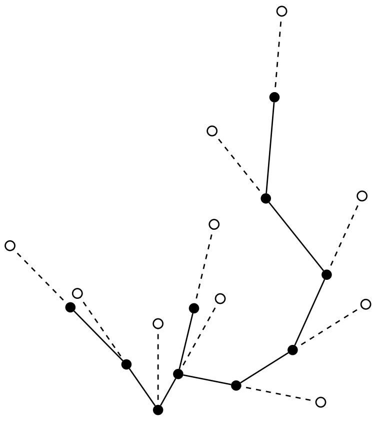
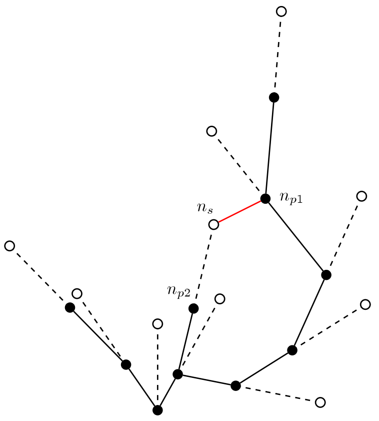
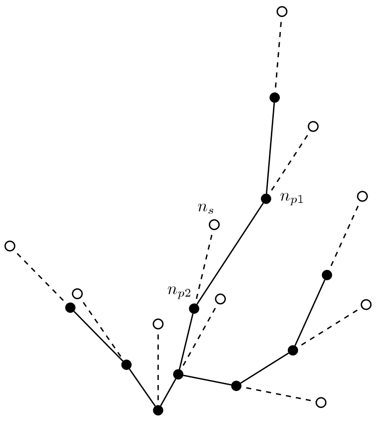

(b) RRT* tree after insertion of the new node into the primary space and the corresponding node into the secondary space.

Fig. 13. Insertion of new nodes (Sampling) into the RRT* tree with *SNN* space. The red filled node indicate a randomly chosen sample. The red line indicates the nearest neighbor correspondence in the secondary space while the red dashed line refers to the related node in the primary space. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) Find nearest neighbor in primary space for each node in secondary space.



(b) Reconnect corresponding nodes of each found pair in primary space and update secondary nodes.

Fig. 14. Reconnection of nodes in the RRT* tree with *SNN* .

6. Evaluation and experiments

Three scenarios have been selected to evaluate different aspects of the proposed navigation and motion planning approach. All experiments have been conducted with the reconfigurable wheel-driven snake-like robot KAIRO 3, introduced in Section 3.

6.1. Scenario 1: uneven ground

The first scenario aimed to evaluate the joint angle planning step. Especially, the roll angle changes to compensate different height levels between the left and right wheel are examined when driving over uneven ground. Therefore, rough stones have been arranged randomly inside a square of two meters, as illustrated in Fig. 15 (top). KAIRO 3 was placed in front of the area and asked to cross this area by a given target pose. First of all the robot automatically took 3-D scans of the environment, created a map and planned a joint angle path to the target pose. While moving along the path, the roll angle and height[13] of the first joint module

[13] The height is measured in the center point of the first drive module which is 0.13 m over ground.

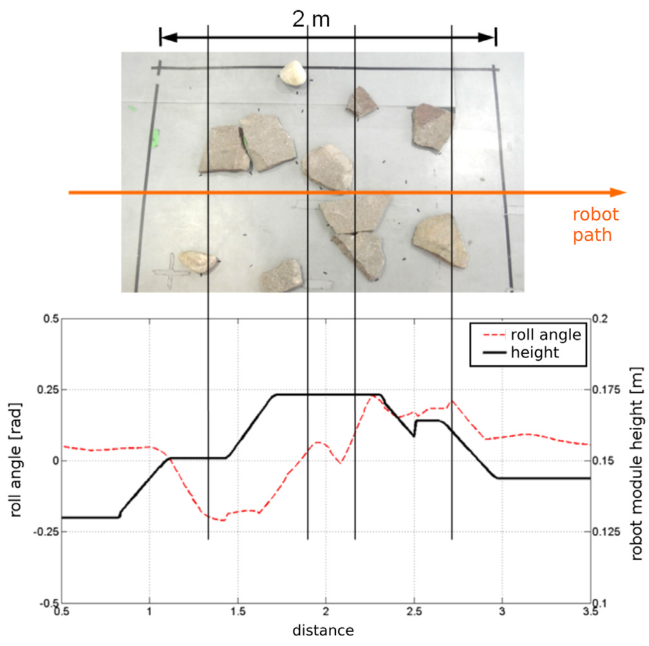
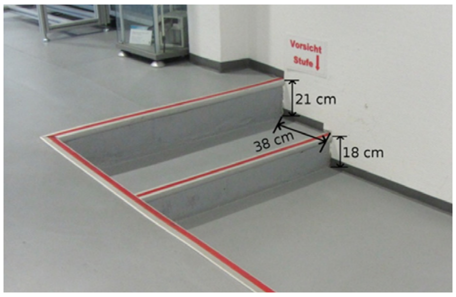
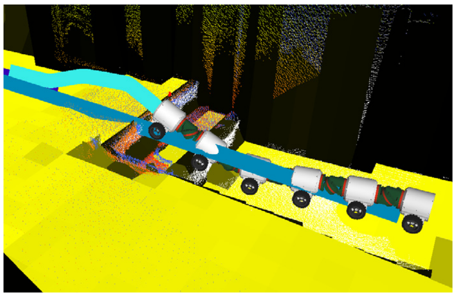


Fig. 15. Evaluation of the roll angle and height adjustment to keep ground contact with the wheels while moving in rough and uneven terrain.



(a) Stair climbing scenario: Stairs with two steps of different heights and a depth of 38 cm.



(b) Visualization of the internal environment representation: The generated *PlexMap* with coarse, precise and joint angle trajectories.

Fig. 16. Evaluation of the climbing abilities to overcome obstacles of different heights and depths using the navigation and motion planning method for snake-like robots.

is measured and analyzed in Fig. 15. Comparing the plotted graphs with the overlaying top view of the area, the turning points of the roll angle graph matches with the obstacles. This means KAIRO 3 lift up the left wheel (roll angle is getting negative) to keep ground contact with both left and right wheel while moving over the stones. Also the height of the module center point increases slightly. Subsequently, the roll angle goes back to nearly zero and the height increases since both wheels are on top of stones. The left wheel moves down first, while the right wheel also moves over the second stone on the right before moving down. The whole movement was repeated by all following modules until KAIRO 3 reached the target point.

6.2. Scenario 2: climbing stairs

The stair climbing scenario, illustrated in Fig. 16a, is used to evaluate the capability in overcoming stairs autonomously with our navigation and motion planning approach. KAIRO 3 was placed about 1 m in front of the steps and a target pose 4 m straight ahead was given manually. Similar to the previous scenario, the robot automatically took 3-D scans, created a map and planned a coarse, precise and joint angle trajectory as shown in Fig. 16b. In the scanning phase, the previously mentioned *Mapping Maneuver* is executed to lift up KAIRO 3's laser scanner module vertically. The upper position, which is shown on the top right image in Fig. 17, results in a much better overview of the environment.

Fig. 17 also illustrates the entire process of mapping, planning and moving along the path, which was executed fully autonomously. Evaluating the repeatability and robustness of our approach, five experiments has been carried out. KAIRO 3 was able to reach the target pose in four of the five experiments without any assistance from a user. One experiment was prematurely aborted due to a localization error which occurred due to a slippage of the robot. Most of the robot odometry and localization discrepancies are automatically corrected after another scan was performed. However, large slippages between two scans may lead to an invalidation of the planned path. Execution times for the several planning steps, shown in Table 1, have been averaged over all five experiments. The planning runs on a i5-Processor with 3.3 GHz and 4 GB RAM.

Table 1
Average calculation times for the several planning steps.

Planning step	Average calculation time
Coarse Planning	5.5 s
Precise Planning	4.5 s
Joint Angle Planning	2.0 s
Total	12.0 s

6.3. Scenario 3: changing robot configurations

The third scenario aimed to evaluate the planning capabilities regarding variations of the robot configuration. With respect to KAIRO 3 this means varying the amount of modules. Fig. 18 (top left) illustrates the prepared scenario consisting of a 54 cm high step next to a 6 m long ramp. Two different configurations of KAIRO 3 have been evaluated: A long configuration consisting of eleven modules and a short with only five modules. In each experiment KAIRO 3 was placed about 1.5 m in front of and facing towards the large step. Similar to the previous scenarios, the mapping and planning has been conducted for a target point in 8 m distance ahead of the robot and on top of the ramp.

The resulting trajectories for the long configuration are visualized on the top left image of Fig. 18. As expected the joint angle path moves up the step with a small overshoot at the step corner. This overshoot is necessary to keep the wheel ground contact all the time since the trajectory is encoded as poses for the center point of the joint modules. The whole process of climbing up the step is shown in the image sequence in Fig. 18.

With the shorter configuration of five modules, the planning system successfully found a path to the target pose, shown in Fig. 19 (left), using a in advance created map. Since the shorter robot is incapable of climbing up the step, the planner performs a U-turn to reach the lower start of the ramp. Due to the small slope the robot is able to reach the target pose but the path is much longer than the direct movement thought the step. Furthermore, Fig. 19 (right) shows that the kinematics constraints for nonholonomic robots are considered correctly by the RRT* algorithm using the *SNN* extension. Measurements of the added samples indicate a reduction of dropped samples by factor five till ten depending on the start and target poses.

7. Conclusions and future work

This article introduced a navigation and motion planning method for reconfigurable snake-like robots. A new adaptive variation of the RRT* algorithm was combined with a joint angle planner to generate complex motions for wheel-driven snake-like robots using 6-D trajectories. The resulting planning system was implemented in three stages for coarse, precise and joint angle planning. The planning time was kept independent from the amount of modules due to a reduced planning space. This was achieved by planning 6-D poses for the first module of the robot combined with the *follow-the-leader* approach applied for motion execution. Concerning the nonholonomic character of snake-like robots, the *Secondary Nearest Neighbor Space* extension for the RRT* has been developed to improve the connectivity of the search tree and to gain better planning results and speed. Considering different robot

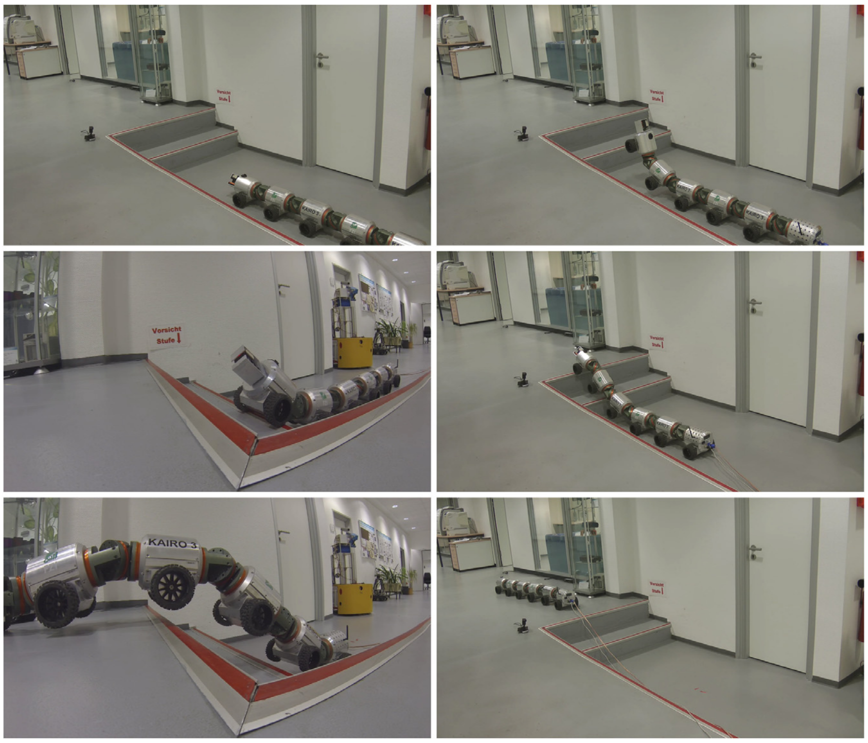


Fig. 17. Sequence showing KAIRO 3 climbing up the stairs of Scenario 2 (Video: https://www.youtube.com/watch?v=orWmDO13k9Y).

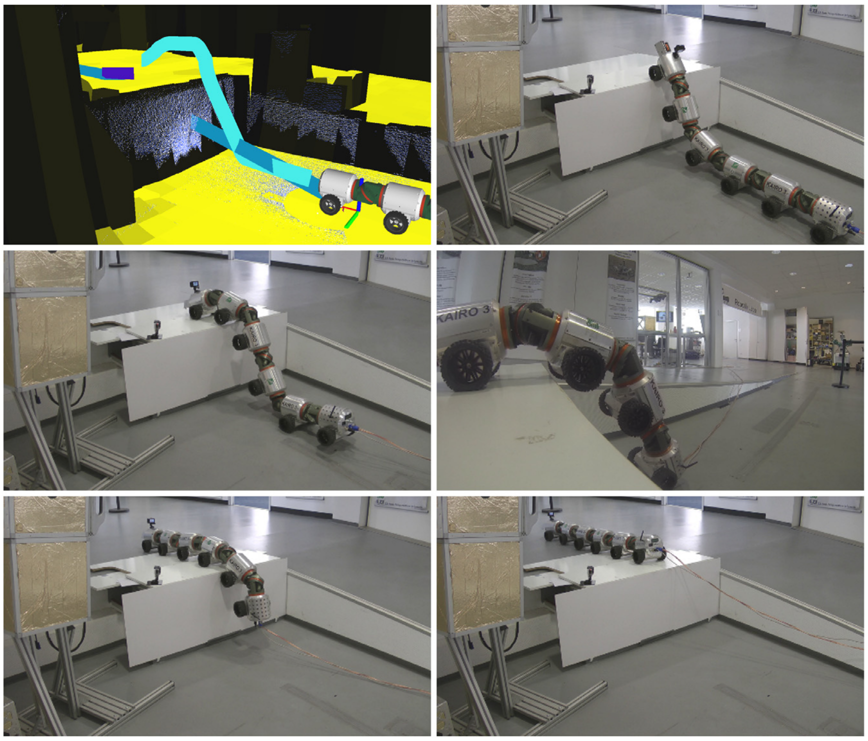


Fig. 18. Sequence showing KAIRO 3 with eleven modules climbing up a 54 cm high step autonomously. Top left image: internal representation of the environment showing point clouds observed with the laser sensor, the *PlexMap* and planned paths at different granularity. Other images: The real robot while proceeding over the obstacle.

configurations, varying in the amount of modules, the adaptive planning system generate trajectories regarding the robot capabilities. The implementation of the proposed approach has been evaluated extensively with KAIRO 3 in three different scenarios. The obtained results demonstrate quite clearly that KAIRO 3 is able to move up stairs and steps autonomously, even if the obstacles are much higher than the robot itself. Our proposed navigation and motion planning along with the extended motion execution of the *Adaptive Control* results in a robust control system for reconfigurable wheel-driven snake-like robots. With regard to our primary

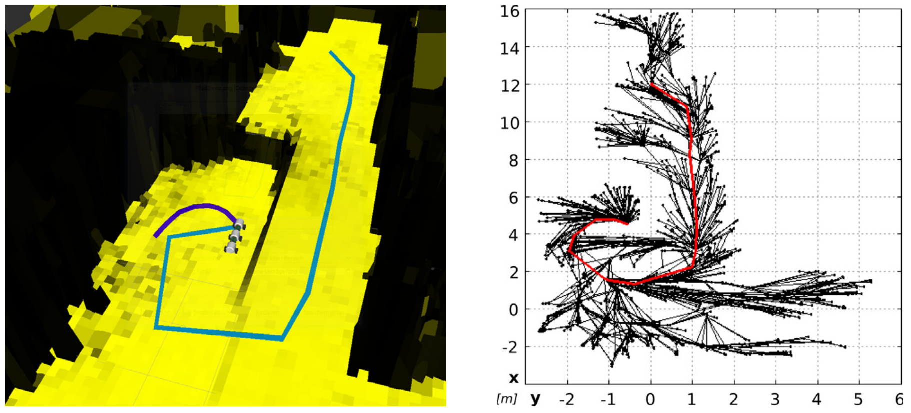


Fig. 19. Generated trajectories for the short configuration of KAIRO 3 are visualized on the left and the RRT* tree build during the coarse planning is shown on the right.

goal, we can state that the autonomy, flexibility and human independence are on a very high level, since the robot is able to reach one or multiple user defined target poses autonomously without further human interaction.

Despite the achieved results, we work on further improvements and investigations. Currently, the utilization of more sophisticated constraints likewise considering reconfiguration costs is under investigation to ensure correct planning for multi-robot systems. In order to prevent unstable and critical situations like the robot is falling over due to unfavorable mass distribution, the existing stability calculation will be integrated into the planning approach. To further speed up the planning process we intend to switch from the RRT*-based approach to the newly introduced RRT*-Connect algorithm [43].

References

[1] A.A. Transeth, P. Liljeback, K.Y. Pettersen, Snake robot obstacle aided locomotion: An experimental validation of a non-smooth modeling approach, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2007, pp. 2582–2589. http://dx.doi.org/10.1109/IROS.2007.4399175.

[2] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in: IEEE Int. Conf. on Robotics and Automation, ICRA, vol. 2, 1985, pp. 500–505. http://dx.doi.org/10.1109/ROBOT.1985.1087247.

[3] J. Ziegler, C. Stiller, Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS, 2009. http://dx.doi.org/10.1109/IROS.2009.5354448.

[4] M. Likhachev, G. Gordon, S. Thrun, ARA*: Anytime A* with provable bounds on sub-optimality, in: In Advances in Neural Information Processing Systems, NIPS, MIT Press, 2003.

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

[6] J.J. Kuffner, S.M. LaValle, RRT-connect: an efficient approach to single-query path planning, in: IEEE Int. Conf. on Robotics and Automation, ICRA, vol. 2, 2000, pp. 995–1001.

[7] S. Karaman, E. Frazzoli, Incremental sampling-based algorithms for optimal motion planning, in: Proc. of Robotics: Science and Systems II, Philadelphia, PA, USA, 2006.

[8] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Robot. Res. 30 (7) (2011) 846–894. http://dx.doi.org/10.1177/0278364911406761.

[9] J.-C. Latombe, Robot motion planning: edition en anglais, in: The Springer International Series in Engineering and Computer Science, Springer, 1991. URL https://books.google.de/books?id=Mbo_p4-46-cC.

[10] S.M. LaValle, Planning Algorithms, Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[11] P. Sanz, Robotics: modeling, planning, and control (Siciliano, B. et al.; 2009) [On the Shelf], IEEE Robot. Autom. Mag. 16 (4) (2009) 101–101. http://dx.doi.org/10.1109/MRA.2009.934833.

[12] S.A.M. Coenen, Motion Planning for Mobile Robots — A Guide. URL http://repository.tue.nl/778545.

[13] D. Tsakiris, M. Sfakiotakis, A. Menciassi, G. la Spina, P. Dario, Polychaete-like undulatory robotic locomotion, in: IEEE Int. Conf. on Robotics and Automation, ICRA, 2005, pp. 3018–3023. http://dx.doi.org/10.1109/ROBOT.2005.1570573.

[14] Z. Bayraktaroglu, A. Kilicarslan, A. Kuzucu, Design and control of biologically inspired wheel-less snake-like robot, in: IEEE/RAS-EMBS Int. Conf. on Biomedical Robotics and Biomechatronics, 2006, pp. 1001–1006. http://dx.doi.org/10.1109/BIOROB.2006.1639222.

[15] J. Conradt, P. Varshavskaya, Distributed central pattern generator control for a serpentine robot, in: Int. Conf. on Artificial Neural Networks, ICANN, 2003.

[16] A.M. Bloch, P.S. Krishnaprasad, J.E. Marsden, R.M. Murray, Nonholonomic mechanical systems with symmetry, Arch. Ration. Mech. Anal. 136 (1996) 21–99.

[17] G. Chirikjian, J. Burdick, An obstacle avoidance algorithm for hyper-redundant manipulators, in: IEEE Int. Conf. on Robotics and Automation, ICRA, vol. 1, 1990, pp. 625–631. http://dx.doi.org/10.1109/ROBOT.1990.126052.

[18] S. Hirose, M. Mori, Biologically inspired snake-like robots, in: IEEE Int. Conf. on Robotics and Biomimetics, ROBIO, 2004, pp. 1–7. http://dx.doi.org/10.1109/ROBIO.2004.1521742.

[19] G. Chirikjian, J. Burdick, The kinematics of hyper-redundant robot locomotion, IEEE Trans. Robot. Autom. 11 (6) (1995) 781–793. http://dx.doi.org/10.1109/70.478426.

[20] J.B. Grzegorz Granosik, Malik G. Hansen, The omniTread serpentine robot for industrial inspection and surveillance, Ind. Robot.: Int. J. (2005). http://dx.doi.org/10.1108/01439910510582264.

[21] G. Granosik, K. Mianowski, M. Pytasz, Wheeeler–hypermobile robot, in: Research and Education in Robotics EUROBOT 2008, vol. 33, Springer, Berlin, Heidelberg, 2009, pp. 68–83. http://dx.doi.org/10.1007/978-3-642-03558-6_7.

[22] T. Kamegawa, T. Yamasaki, F. Matsuno, Evaluation of snake-like rescue robot "KOHGA" for usability of remote control, in: IEEE Int. Workshop on Safety, Security and Rescue Robotics, SSRR, 2005, pp. 25–30. http://dx.doi.org/10.1109/SSRR.2005.1501269.

[23] L. Pfotzer, S. Ruehl, G. Heppner, A. Roennau, R. Dillmann, KAIRO 3: A modular reconfigurable robot for search and rescue field missions, in: IEEE Int. Conf. on Robotics and Biomimetics, ROBIO, 2014, pp. 205–210. http://dx.doi.org/10.1109/ROBIO.2014.7090331.

[24] K.-U. Scholl, Konzeption und Realisierung einer Steuerung für vielsegmentige, autonome Kanalroboter, Verl. für Wissenschaft und Kultur WiKu-Verl. Stein, Berlin, 2003.

[25] S. Yamada, H. Hirose, (2006) Development of Practical 3-Dimensional Active Cord Mechanism ACM-R4.

[26] G. Granosik, M. Pytasz, Control methods for wheeeler the hypermobile Robot, in: K. Kozowski (Ed.), Robot Motion and Control, in: Lecture Notes in Control and Information Sciences, vol. 422, Springer, London, 2012, pp. 27–37. http://dx.doi.org/10.1007/978-1-4471-2343-9_3.

[27] B. Murugendran, A. Transeth, S. Fjerdingen, Modeling and path-following for a snake robot with active wheels, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS, 2009, pp. 3643–3650. http://dx.doi.org/10.1109/IROS.2009.5353886.

[28] C. Altafini, Some properties of the general n-trailer, Internat. J. Control 74 (4) (2001) 409–424.

[29] T. Kano, A. Ishiguro, Obstacles are beneficial to me! Scaffold-based locomotion of a snake-like robot using decentralized control, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2013, pp. 3273–3278. http://dx.doi.org/10.1109/IROS.2013.6696821.

[30] T. Kamegawa, R. Kuroki, M. Travers, H. Choset, Proposal of EARLI for the snake robot's obstacle aided locomotion, in: IEEE Int. Symposium on Safety, Security, and Rescue Robotics, SSRR, 2012, pp. 1–6. http://dx.doi.org/10.1109/SSRR.2012.6523889.

[31] P. Liljeback, K.Y. Pettersen, Ø. Stavdahl, J.T. Gravdahl, Experimental investigation of obstacle-aided locomotion with a snake robot IEEE Trans. on Robotics 27 (4) (2011) 792–800. http://dx.doi.org/10.1109/TRO.2011.2134150.

[32] A. Roennau, G. Heppner, M. Nowicki, R. Dillmann, LAURON V: A versatile six-legged walking robot with advanced maneuverability, in: IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics, AIM, 2014, pp. 82–87. http://dx.doi.org/10.1109/AIM.2014.6878051.
[33] K. Uhl, M. Ziegenmeyer, MCA2—An extensible modular framework for robot control applications, in: Advances in Climbing and Walking Robots: Proceedings of the 10th Int. Conf. on Climbing and Walking Robots, CLAWAR, World Scientific, 2007, pp. 680–689.
[34] K. Regenstein, Modulare, verteilte Hardware-Software-Architektur für humanoide Roboter (Ph.D. thesis), FZI Karlsruhe, 2010.
[35] L. Pfotzer, S. Bohn, C. Birkenhofer, R. Dillmann, Biologically-inspired locomotion with the multi-segmented inspection robot KAIRO-II, in: Proc. of CLAWAR2011, 14th Int. Conf. on Climbing and Walking Robots, 2011, p. 181. http://dx.doi.org/10.1142/8305.
[36] C. Birkenhofer, Adaptive Steuerung eines mehrsegmentigen Inspektionsroboters (Ph.D. thesis), FZI Karlsruhe, 2010.
[37] J. Oberlander, S. Klemm, G. Heppner, A. Roennau, R. Dillmann, A multi-resolution 3-D environment model for autonomous planetary exploration, in: IEEE Int. Conf. on Automation Science and Engineering, CASE, 2014, pp. 229–235. http://dx.doi.org/10.1109/CoASE.2014.6899331.
[38] L. Pfotzer, J. Oberlaender, A. Roennau, R. Dillmann, Development and Calibration of KaRoLa, a compact, high-resolution 3D laser scanner, in: IEEE Int. Sym. on Safety, Security, and Rescue Robotics, SSRR, 2014, pp. 1–6. http://dx.doi.org/10.1109/SSRR.2014.7017677.
[39] S.M. LaValle, Rapidly-Exploring Random Trees: A New Tool for Path Planning, Tech. rep., Department of Computer Science, Iowa State University, 1998.
[40] S.M. LaValle, J.J. Kuffner, Randomized kinodynamic planning, Int. J. Robot. Res. 20 (5) (2001) 378–400.
[41] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is "nearest neighbor" meaningful? in: Int. Conf. on Database Theory, 1999, pp. 217–235.
[42] A. Qureshi, S. Mumtaz, K. Iqbal, B. Ali, Y. Ayaz, F. Ahmed, M. Muhammad, O. Hasan, W.Y. Kim, M. Ra, Adaptive potential guided directional-RRT, in: IEEE Int. Conf. on Robotics and Biomimetics, ROBIO, 2013, pp. 1887–1892. http://dx.doi.org/10.1109/ROBIO.2013.6739744.
[43] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J.M. Zöllner, R. Dillmann, RRT*-Connect: Faster, asymptotically optimal motion planning, in: IEEE Int. Conf. on Robotics and Biomimetics, ROBIO, 2015.

Lars Pfotzer is working as a research scientist at the department Interactive Diagnosis and Service Systems (IDS) at the FZI Research Center for Information Technology in Karlsruhe. He studied Computer Science at the Karlsruhe Institute of Technology (KIT) till 2009 with specialization in robotics and automation as well as embedded systems. His research topics include locomotion and navigation of reconfigurable robots with a focus on modular snake-like inspection robots. Additionally he is working on optimization algorithms for generating configurations of modular and reconfigurable systems.

Sebastian Klemm is working as research scientist at the department Technical Cognitive Systems (TKS) at the FZI Research Center for Information Technology in Karlsruhe. He studied Computer Science at the Karlsruhe Institute of Technology (KIT) with focus on communications engineering, anthropomatics and robotics, and received his Diploma degree in 2013. His research interest is navigation of autonomous systems in unknown, unstructured and dynamic environments with focus on motion planning. He also addresses the field of automated driving and parking of intelligent vehicles.

Arne Roennau is the department manager of the department Interactive Diagnosis and Service Systems (IDS). He studied Electrical Engineering and Information Technology at Karlsruhe Institute of Technology (KIT), specialized in the fields of feedback control, automation and robotics. Since November 2008 he has been working in the IDS department and has been active as department manager since 2011 as well as head of the FZI Living Lab Service Robotics. His main fields of research are robot motion control, 3D perception and environment modeling and the design and construction of new service and field robots.

J. Marius Zöllner studied Computer Science with special focus on artificial intelligence and robotics at the University of Karlsruhe where he also received his Dr.-Ing. degree (Ph.D.) in 2006. From 1999 he worked with FZI Research Center for Information Technology where he became division manager in 2006. Since 2008 he is Professor for Applied Technical Cognitive Systems at the KIT, Karlsruhe Institute of Technology and Director at the FZI. Current research activities are focusing on cognitive cars and service robotics. Since 2012 he is also member of the executive board of the FZI.

Ruediger Dillmann is a Professor of the Department of Computer Science at the Karlsruhe Institute of Technology (KIT), Germany. He is speaker of the Institute for Anthropomatics (IFA) and leads the Humanoids and Intelligence Systems Laboratories (HIS). At FZI he is the Director of the department Interactive Diagnosis and Service Systems (IDS). As a leader of these institutes he supervises several research groups in the areas of robotics with special interests in intelligent, autonomous and mobile robotics, humanoids, machine learning, machine vision and human–machine interaction.