2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Fast on-board motion planning for modular robots

Vojtěch Vonásek[1], *Student member, IEEE*, Lutz Winkler[2], Jens Liedke[2]
Martin Saska[1], *Member, IEEE*, Karel Košnar[1] and Libor Přeučil[1], *Member, IEEE*

***Abstract*— Modular robots, which are systems made of many robotic modules, can utilize various types of locomotion. Different approaches can be used to generate these basic motion skills — motion primitives. To move in a complex environment, several motion primitives are needed and a mechanism to switch them is required. This can be realized using a high-level motion planning. To enable autonomous operation of modular robots equipped with limited computational resources, it is necessary to generate the motion plans on-board, i.e., without external computers. In this paper, we propose a novel simplified motion model of a modular robot, which allows the robot to employ the motion planner as a fast on-board replanner. The proposed approach has been verified both in simulations as well as with real robots.**

I. Introduction

Modular robots consist of many robotic modules, which can be connected to form robots of various shapes (example of a snake-like modular robot is depicted in Fig. 1). Modules that support autonomous connection/reconnection can be formed into self-reconfigurable robots with the possibility to change their structure to meet operational demands. In comparison to conventional fixed-shaped robots, this may bring additional abilities in applications like space exploration, search and rescue missions or object manipulation. A survey of existing modular robots can be found in [8].

Three main concepts can be utilized to achieve motion of the modular robots: a) wheeled locomotion; b) reconfiguration; and c) joint-control. In wheeled locomotion, wheels [1], belts or even screwdrivers [7] are utilized. In the concept of motion through reconfiguration, a module is disconnected from the robot, it moves to a target position, where it connects back to the organism [9]. A widely used type of locomotion is the joint-control approach, where a motion is achieved by controlling joints between the connected modules. This approach is considered in this paper.

Many approaches have been proposed to control the joints of the modular robots to achieve a desired behavior, such as Central Pattern Generators (CPG) [2]. Although these approaches can efficiently generate various types of locomotion like crawling or walking, they provide rather local locomotion in a vicinity of a robot. To achieve a global and possibly distant goal, the local motions need to be combined using a high-level motion planner.

In our previous work [14], we have proposed a high-level RRT-MP motion planner (Rapidly Exploring Random Tree with Motion primitives), that utilizes a set of predefined motion primitives. The RRT-MP employs a physical simulation to estimate robot's response to a motion primitive. As the evaluation of physical simulation is computationally intensive, it needs to be run on a powerful computer. This precludes to utilize the original RRT-MP on-board, as the computational resources of modular robots are typically limited.

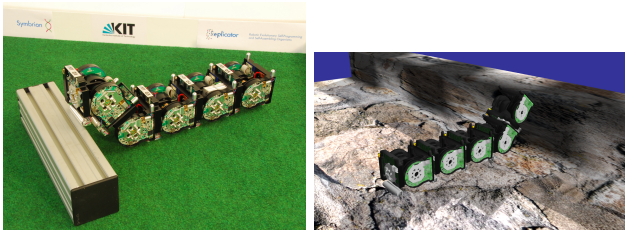


Fig. 1. A snake robot made of five KaBot modules developed within Symbrion/Replicator projects (left) with its simulated version (right).

In this paper, we present a novel approach to model motion of modular robots. Instead of using a precise physical simulation to evaluate motions of a robot, the proposed approach describes the motion using simple equations, which can be evaluated quickly. This allows the robot to generate motion plans on-board, without need to utilize an external computer. Consequently, the on-board planner can be used as a feedback replanner. This increases reliability of the motions, as a new plan can be easily generated when a robot deviates from an expected position.

The paper is organized as follows. The related work is described in the next section, and the used notation in Section III. The RRT-MP motion planner is briefly described in Section VI. The novel motion model, called Simplified Motion Model (SMM) is described in Section VII. The performance of the RRT-MP with the novel SMM model was experimentally verified. The results of the experiments are described in Section VIII. Discussion can be found in Section IX.

II. Related work

Motion planning for modular robots is a challenging task especially due to many degrees of freedom (DOF). Moreover, the motion of the individual modules is constrained by the mechanical connections between them as well as by contacts with the underneath terrain. Motion planning for such a system can be solved using sampling-based approaches like Rapidly Exploring Random Tree (RRT) [6]. The idea of the sampling-based methods is to create a roadmap of the

[1]Dept. of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Technicka 2, 166 27, Prague 6, Czech Republic; {vonasek,saska,kosnar,preucil}@labe.felk.cvut.cz
[2]Institute for Process Control and Robotics (IPR), Engler-Bunte-Ring 8, D-76131 Karlsruhe, Germany; {winkler,liedke}@kit.edu

configuration space of a robot using a randomized sampling. Then, a motion in the workspace can be found as a path in the roadmap.

In the RRT algorithm, which is utilized in this paper, a motion model of a robot is used to generate new samples for the roadmap. This requires to generate suitable control signals for the robot. While the control signals can be easily generated for simple robots (e.g. differential drive mobile robots), the generation of the control signals for modular robots is more challenging. To generate suitable input signals, methods for locomotion generation can be utilized.

Widely used concept for motion generation of modular robots are Central Pattern Generators (CPGs), which produce periodic control signal for the actuators. By changing parameters and coupling of the CPGs, various gaits like crawling or caterpillar-like motions can be achieved. The CPGs are frequently used for locomotion of modular robots; we refer to [2] for a detailed review. A decentralized approach for locomotion generation is based on rules [12]. In this concept, the actuators are controlled using cyclic functions. After a module completes a fraction of the period, it sends a message to its neighbors, which synchronize the neighboring modules. Another decentralized signal synchronization can be based on Artificial Hormones [11].

The control signals provided by the above mentioned locomotion generators can be considered as motion primitives. The primitives represent basic motion skills of the robot, like 'go-forward' or 'turn-left'. To move a robot into a distant location in a complex environment, more motion primitives are needed and they need to be switched. For example, a robot moving in complex scenarios containing both planar-like areas as well as areas with rough terrain needs to be equipped with at least two motion primitives. To visit a distant goal, the robot needs to utilize suitable primitive on the planar-like part of the environment and it needs to switch to a different one when it enters the rough terrain. The sequence of motion primitives, needed to traverse the environment from an initial configuration to a goal configuration, can be found using motion planning.

III. Preliminaries

Let $q = (x, y, z, \alpha, \beta, \gamma, a_1, \ldots, a_{n-1})$ denote a configuration of a modular robot made of n modules connected to a structure without cycles, where (x, y, z) and (α, β, γ) denote position and rotation of a pivot module resp. and a_i are angles of the joints. The set of all configurations is the configuration space $\mathcal{C}$. Free configurations, where the robot does not collide with any obstacle and satisfies all kinematic constraints, form the subset $\mathcal{C}_{free} \subseteq \mathcal{C}$. In the joint-control approach, the motion of the robot is achieved by controlling the angles a_i. We assume, that each module is able to reach a desired angle, therefore, the control signal $\mathbf{u}(t) = (a_1(t), \ldots, a_{n-1}(t))$ of desired angles can be used to control the whole robot.

IV. Motion planning

The task of the motion planning is to generate a feasible trajectory between an initial configuration $q_{init} \in \mathcal{C}_{free}$ and a goal configuration $q_{goal} \in \mathcal{C}_{free}$. In this paper, RRT-based planning [6] is utilized. The RRT algorithm randomly samples the configuration space $\mathcal{C}$ and builds a tree $\mathcal{T}$ of feasible configurations. The tree is rooted in the initial configuration q_{init} and it is extended in each iteration as follows. First, a random sample $q_{rand} \in \mathcal{C}$ is generated. The nearest node q_{near} in the tree is found and expanded to obtain a set of free configurations reachable from q_{near}. From this set, a configuration q_{new} with corresponding control signal $\mathbf{u}$ that is nearest to q_{rand} is selected and stored into the tree. The algorithm terminates if the goal region defined by the radius d_{goal} is approached. The main loop of the RRT algorithm is listed in Alg. 1.

Algorithm 1: RRT algorithm.

Input: initial configuration q_{init}, goal configuration q_{goal}, goal region d_{goal}, maximum number of iterations k

Output: path to the goal configuration or failure

```
1  T.addNode(q_init);
2  for iteration = 1 ... k do
3      q_rand = generate random sample in C;
4      q_near = nearestConfiguration(T,q_rand);
5      q_new, u = expandConfiguration(q_near, q_rand);
6      if q_new ≠ NULL then
7          T.addNode(q_new, u);
8          T.addEdge(q_near,q_new);
9          if distance(q_new, q_goal) < d_goal then
10             return path in the tree from q_init to q_goal;
11         end
12     end
13 end
14 // trajectory was not found during k iterations;
15 return failure;
```

In the expansion step, which is a crucial part of the RRT planners, suitable control signals need to be generated. In the case of robots with few control inputs (e.g. a mobile robot), values of control inputs can be discretized and all combinations of these values can be examined. This cannot be used in the case of modular robots with many actuators to be controlled. In [13], the control inputs of modular robots are randomly generated in each expansion step. As the generated inputs differ in each iteration, they allow the robot to move. However, the resulting motions are clumsy and ineffective. To generate suitable control inputs, the concept of motion primitives can be used.

V. Motion primitives

A motion primitive p is represented by a control signal $\mathbf{u}_p(t)$. The primitives can be realized using the locomotion generators such as CPGs. A robot can be equipped with a library of K primitives, that represent skills necessary to accomplish a given task. For example, the primitives such as 'turn' and 'move' can be satisfactory to move on a plane.

When the robot has to fulfill a different task such a climbing, additional primitives should be used.

For each motion primitive, it is necessary to define preconditions defining whether a primitive can be used. This might depend on the state of robot or on the previously used primitive. Two primitives may be prohibited to be used consecutively e.g. due to possible stability problems [4]. Let $isApplicable(p_1, p_2)$ be true if the primitive p_2 can be applied after applying primitive p_1. Similarly, let us define $isApplicableOnState(q, p)$ to return true if the primitive p can be applied in a configuration q.

VI. Motion planning with motion primitives

In our previous work [14], we have introduced the RRT-MP planner (RRT with Motion Primitives). The main idea of the algorithm is to use a set of motion primitives to expand the configuration tree. The algorithm is listed in Alg. 2. In comparison to basic RRT, the RRT-MP only changes the expansion step, where the provided motion primitives are examined to find a new state q_{new} closest to the random configuration q_{rand}. Before a motion primitive is examined, its preconditions need to be checked.

Each node in the constructed tree holds information about motion primitive used to reach the node, as well as a pointer to the previous node. An example of the constructed tree is depicted in Fig. 2. To retrieve a plan, the closest node in the tree toward q_{goal} is found and the tree is backtracked to the q_{init} node. The resulting plan can be represented both as sequence of motion primitives (or their indexes) or as a sequence of configurations. While the information about utilized motion primitives is necessary to navigate a robot according to the plan, the information about plan's configurations is necessary to check possible deviation of the robot from the plan.

Algorithm 2: expandConfiguration() of RRT-MP

Input: random conf. q_{rand}, conf. to expand q_{near}
Output: configuration reachable from q_{near}

1 $S = \emptyset$;
2 p_{prev} = motion primitive used to reach q_{near};
3 **foreach** $i \in K$ **do**
4 **if** $rand() < r$ **then**
5 **if** $isApplicable(p_{prev}, p_i)$ *and* $isApplicableOnState(q_{near}, p_i)$ **then**
6 $u(t)$ = load i-th motion primitive
7 **end**
8 **else**
9 $u(t)$ = generate random signals
10 **end**
11 q = motionModel$(q_{near}, u(t))$, $t = [0, \Delta t]$;
12 $S = S \cup \{(q, i)\}$;
13 **end**
14 q_{new}, u = closest configuration from S to q_{rand};
15 return (q_{new}, u);

Beside the reduction of the complexity of the expansion step, the utilization of motion primitives brings additional advantages. First, we can assume that the primitives move

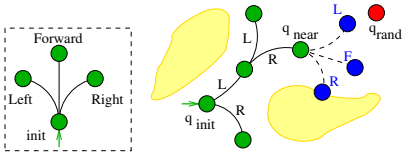


Fig. 2. Example of a configuration tree constructed using three motion primitives. During the expansion step, all three primitives (Left, Right and Forward) are examined and applied from state q_{near}. The tree is then extended by such a state, which is closest to the q_{rand} (Left in this case).

the robot more efficiently, than a randomly generated control input. Therefore, the resulting motions are more fluid. Second, the motion primitive can move the robot to a more distant location, which speeds-up coverage of the configuration space. Consequently, fewer iterations are needed to reach the goal configuration, which speeds-up the planning process. Third, as the motion primitives abstract the planner away from the actuators, the complexity of the planning algorithm is not influenced by the number of the modules in a robot. Another important advantage of planning with motion primitives is the reduction of memory consumption. As the RRT-MP utilizes a predefined set of motion primitives, only their indexes need to be stored in the tree. This requires significantly less number of variables comparing to amount of memory needed to store full control signals.

To preserve the probabilistic completeness of the RRT algorithm, both the predefined motion primitives and random signals can be used in the RRT-MP. If only motion primitives are used, the algorithm is able to move the robot only by steps provided by the primitives. For example, a robot equipped only by 'step-forward' and 'one-step-left' primitives moves on a grid defined by these primitives. As the random signals are used with nonzero probability, the algorithm is able to reach other positions, that are not reachable by the primitives only. Therefore, the random control inputs are used with probability r in Alg. 2.

VII. Simplified motion model

A motion model describes a change of positions of robot's modules after a control signal is applied. It is used in the expansion step to obtain new configurations reachable from q_{near} (line 11 in Alg. 2). Due to kinematic constraints and constraints caused by interactions with the underneath terrain, a precise closed-form motion model of a modular robot cannot be easily derived [10]. Instead, the motion model can be realized using physical simulation [13].

In the RRT-based planners, the motion model is used as a black-box and only the resulting configuration is needed. Therefore, it is sufficient to model only the change of the robot's position without need to detail model intermediate motions. In this paper, we propose to use such a model, which is called Simplified Motion Model (SMM) in the rest of the paper.

We assume, that a motion of a robot controlled by a short motion primitive p can be approximated by a straight line. Then, the motion can be described as a combination of rotation about angle α_p, forward translation to distance

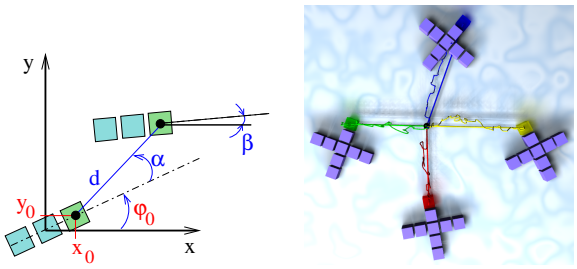


Fig. 3. Example of four motion primitives (each primitive is depicted in different color). The primitives are visualized as curved paths representing motion of robot's pivot module. The straight lines represent the SMM model of these four motion primitives. The robots are placed in the final position after 10s of motion. The black point represents the initial position of the robot.

d_p and rotation to its final heading about angle β_p. The change of robot's height is described by parameter c_p and the changes of joint angles are described by $\delta_{p,i}, i = 1, \ldots, n-1$. Such a motion model $\dot{q} = f(m_p, q)$ can be described as

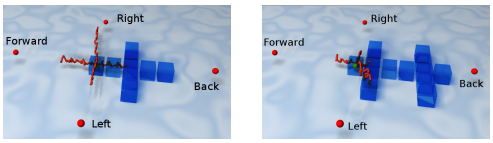
$$\begin{aligned} \dot{x} &= d_p \cos(\varphi + \alpha_p) \\ \dot{y} &= d_p \sin(\varphi + \alpha_p) \\ \dot{z} &= c_p \\ \dot{\varphi} &= \beta_p \\ \dot{a_i} &= \delta_{p,i}, \end{aligned} \tag{1}$$

where $m_p = (d_p, \alpha_p, \beta_p, c_p, \delta_{p,1}, \ldots, \delta_{p,n-1})$ are parameters of the model. The schema of the SMM is depicted in Fig. 3a and an example of a SMM for Quadropod robot is depicted in Fig. 3b.

To parameters m_p of the SMM have to be found for each primitive p. These can be easily estimated as the changes of robot's position and change of its heading. As the duration of motion primitives is not considered in the SMM, it is assumed, that the primitives are always applied over the same time interval. If one needs to apply same primitive over different time periods, the parameters m_p should be derived for each of the considered periods.

The resulting configuration of a robot after applying a motion primitive can also be influenced by the initial state of the robot. This initial state is caused by the previously used primitive. To increase precision of the SMM, the parameters $m_{i,j}$ can be used instead of m_p. Here, parameters $m_{i,j}$ can be derived for each combination of previously used primitive i and actually used primitive j.

The utilization of the fast SMM in the RRT-MP significantly speeds-up the planning process. This allows the robot to use the RRT-MP as an on-board planner. After a plan is created, the robot obtains the planned sequence of the motion primitives. While the robot executes the plan, its position is tracked using a global localization. When the robot deviates from the planned trajectory, a new plan from actual position to the desired goal state can be immediately created. The deviation from the plan can be caused by inaccuracies of the SMM, by an unexpected slippage or imprecise control



Quadropod Lizard

Fig. 4. Visualization of motion primitives found using PSO technique. The primitives are shown as trajectories of the pivot module. The red points represent the virtual goals, which are used to evaluate the fitness function. The robots are placed in their initial positions.

of the robot's joints. Theoretically, a possible inaccuracy of the motion model can be decreased by employing a more detailed model, or even by precise physical simulation. However, such a motion model will require more time to be evaluated, which slows down the planning process. Moreover, neither the alternative can eliminate unexpected control problems. In both cases, the robot can navigate more precisely when a fast replanning is utilized.

VIII. Experimental verification

The proposed system for fast on-board motion planning was verified in simulated as well as real scenarios. The KaBot modular robots developed within Symbrion/Replicator projects [7] are used for the experiments. The KaBot modules are cubes of side length 12 cm and weight ∼1 kg. They provide 2D locomotion using two screwdrives as well as 3D locomotion using a movable hinge. Each module is equipped with four docking mechanisms, camera, IR receivers/transmitters and accelerometers. The data can be processed by the on-board Blackfin CPU. The modules communicate with each other using Ethernet, Bluetooth or Zigbee. The robots are externally localized in the experimental arena using camera-based localization [5].

A. Learning of motion primitives in simulation

For the simulated experiments, two modular robots are used: Lizard (made of $n = 13$ modules) and Quadropod ($n = 9$ modules). The ODE physical engine is used to simulate the modular robots. The locomotion of the robots is generated using sinus CPG, where position of each joint $a_i(t) = A_i \sin(2\pi f_i t + \varphi_i) + B_i$. The parameters $x_p = (A_i, B_i, f_i, \varphi_i)$, $i = 1 \ldots, n-1$ for each motion primitive p are found using Particle Swarm Optimization (PSO) [3] method with 30 particles and 200 generations. The ranges of the parameters are: $A_i = (0, \pi/2)$, $f_i = (0, 2)$ Hz, $\varphi_i = (0, 2\pi)$ and $B_i = (-\pi/2, \pi/2)$. The task of the PSO is to find four primitives p, where $p \in \{$left,forward,right,back$\}$. The primitives should move the robot in a given direction. Therefore, the fitness function is defined as robot's distance to a virtual point in the desired direction. The virtual point is place to distance 5 map units from the initial position of the robot. The fitness function is evaluated after 10 s of simulated motion. Examples of the resulting primitives are depicted in Fig 4. These primitives are used in the following experiments.

B. Influence of conditioned models

The influence of SMM's precision to performance of planning is investigated in the first simulated scenario. Two versions of the SMM are defined: a) the parameters m_p of the SMM are estimated for each motion primitive p; b) the parameters $m_{i,j}$ are estimated for each combination of previously used primitive i and actually used primitive j. The second approach, called Coupled SMM, thus considers influence of the previously utilized primitives.

The size of the simulation environment is 50x25 map units (mu), which is large enough in comparison to the size of each module (0.5 mu). To evaluate performance of the planner, the motion planning should be performed between many different start and goal configurations, which is similar to our target application (search and rescue). Therefore, 126 pairs of start/goal configurations are randomly generated in $\mathcal{C}_{free}$. A trajectory between each start/goal pair is found using the RRT-MP planner with the tested SMM models. The planning between each start/goal pair is repeated 20 times. This schema is repeated for each robot and for both versions of the SMM's parameters. The maximum number of allowed iterations of the RRT-MP (parameter k in Alg. 1) is set to 400. The duration of each motion primitive is 5 s. The experiment is run on Intel Core 2@3.16 GHz under FreeBSD 8 operating system.

After a plan for a start/goal pair is created, the simulated robot navigates according to the plan in an open-loop way. To evaluate quality of the plan, its success rate and distance to goal are measured. The success rate of a pair is defined as a ratio of number of trials where the robot reached the goal configuration to the number of all trials (number of trials is 20). The global success rate is computed as average success rate over all start/goal pairs. The configuration is reached, when the robot approaches it to the distance of 1 map unit or less. The average distance to goal is computed over all start/goal pairs and over all trials.

The results are summarized in Tab. I. These measured values are also statistically compared using t-test. In this table, we do not show runtimes of the RRT-MP planner, because the average runtime is 50 ms, which is negligible in comparison to the time needed to traverse the resulting plans (tens of seconds). The results of the RRT-MP with Coupled model are better (they approach the goal configurations to closer distance and also the success rate is higher).

The success rate is not 100 % for either method. There are two possible reasons. First, feasible plans for some randomly generated start/goal pairs cannot be found because the number of planning iterations is limited. Second, low success rate can be caused by open-loop control of the robot along the plan. Due to slippage or imprecise control, the robot may fail to precisely execute the plan. Therefore, it deviates from the planned trajectory and cannot reach the goal. To improve the success rate, the robots should be navigated to the goal using replanning, which is evaluated in the next section.

TABLE I

PERFORMANCE OF MOTION PLANNING WITH TWO VARIANTS OF SMM'S PARAMETERS (INDICATED BY COUPLED COLUMN).

	Coupled	Quadropod	Lizard
Distance [mu]	0	8.83(11.54)	8.59(15.12)
	1	5.37(4.46)	5.25(4.89)
	p-value	< 0.001	0.015
Success rate [%]	0	51.71(29.01)	52.90(36.76)
	1	64.05(25.59)	63.17(32.77)
	p-value	< 0.001	0.020

C. Navigation of the robots using replanning

In the second simulated experiment, we verify performance of the RRT-MP planner utilized as a fast replanner. The position of a robot is measured during the navigation. When the robots deviates from the planned trajectory to more than 2 map units, the RRT-MP is called to generate a new plan from actual position to the goal.

The results are summarized in the Tab. II. The utilization of replanning leads to higher success rates and it better approaches the goals. As this experiments are performed on the same environment and with the same start/goal pairs as in the previous one, we can compare the achieved results with success rates in Tab. I. The success rates achieved with replanning are further increased compared to success rates achieved with the Coupled model. The fact, that success ratio is not 100 % even when replanning is allowed, is caused by randomly generated start/goal pairs. To further increase the success rate, the number of allowed iterations of the RRT-MP should be increased.

TABLE II

DIFFERENCE BETWEEN NAVIGATION OF THE ROBOT ACCORDING TO SINGLE PLAN (REPLAN=0) AND NAVIGATION WITH REPLANNING (REPLAN=1).

	Replan	Quadropod	Lizard
Distance [mu]	0	5.25 (4.39)	5.31 (5.02)
	1	2.14 (0.74)	2.03 (0.69)
	p-value	< 0.001	< 0.001
Time [s]	0	0.02 (0.00)	0.02 (0.00)
	1	0.03 (0.01)	0.03 (0.01)
	p-value	< 0.001	< 0.001
Success rate [%]	0	64.56 (28.09)	64.33 (32.14)
	1	86.83 (13.21)	89.92 (11.94)
	p-value	< 0.001	< 0.001

D. Motion planning with real robots

The performance of the RRT-MP planner was verified in a scenario with a modular robot made of five KaBot modules. The motion primitives created in the simulation are utilized for real robot. To estimate the parameters of the SMM, the motion primitives are repeated 10 times on the robot. The parameters d_p, α_p and β_p are estimated using global position of the robot. The change of angles $\delta_{p,j}$, where $j = 0, \ldots, 4$, of the modules are measured using internal sensors. The

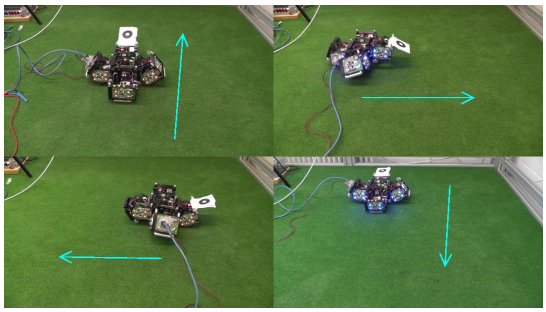


Fig. 5. Snapshots from identification of SMM's parameters. Each of the four motion primitives is repeated 10 times on the robots. The arrows show the direction of movement.

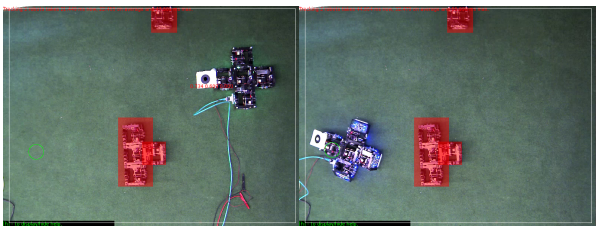


Fig. 6. Snapshots from the navigation to the desired area (depicted by green circle). The red areas represent obstacles. The full motion took $\sim$ 5 minutes.

snapshots of the identification of SMM's parameters with real robots are depicted in Fig. 5 [1].

The task of the robot is to move in the arena to a given target position. The snapshots from the experimental arena are depicted in Fig. 6. The RRT-MP planner is run on-board, i.e., on the robot's BlackFin@550 MHz PC. The planning takes $\sim$ 1 s on this CPU. The experiment was repeated 10 times. Average length of the plan was 11 steps. Execution of one primitive takes 30 seconds, therefore the navigation to goal position takes about 5 mins. The planning time is thus negligible in comparison to this time. New plan is created, if the robot deviates from its planned position to more than 18 cm, which is 1.5x size of the robot.

IX. Discussion

The experiments have shown, that the proposed simplified motion model for modular robots can be successfully utilized in the motion planning task. The SMM can be evaluated quickly, which is necessary for the on-board planning. Possible inaccuracies of the SMM can be compensated by the fast replanning.

In Tab. I we can see that utilization of the Coupled SMM increases the success rate from 51.71% to 64.05% for the Quadropod robot and from 52.9% to 63.17% for the Lizard robot respectively. The difference between the success ratios is 12.34 percentage points (pp) for Quadropod and 10.26 pp for Lizard respectively. The Coupled SMM, which considers influences of previously applied motion primitive, thus increased the success ratio. By analyzing data in Tab. II, we can see, that utilization of replanning increases the success ratio about 22.27 pp for Quadropod and about 25.59 pp for Lizard robot respectively. We can conclude that allowing replanning during the navigation increases the reliability of the navigation more than utilization of the Coupled SMM.

[1] Video can be downloaded at: http://www.youtube.com/watch?v=fCy3grSRC9k

X. Conclusion

In this paper, we have described a novel fast motion planning for modular robots. The proposed planner, called RRT-MP, utilizes the concept of motion primitives, that are implemented using a locomotion generator. To enable motion planning on-board, i.e., on computers available on modular robots, we proposed a novel motion model, called SMM (Simplified Motion Model) that approximates robot's motions. The proposed motion model can be easily evaluated, which enables the RRT-MP to be run on a computer with limited resources.

XI. Acknowledgments

The work of M. Saska was supported by GACR under grant no. GPP103/12/P756, the work of other authors was supported by TACR grant No. TE01020197. The experiments have been run using Grid Infrastructure Metacentrum (project No. LM2010005). The authors would like to thank reviewers for their valuable comments and suggestions.

References

[1] F. Hou, N. Ranasinghe, B. Salemi, and Wei-Min Shen. Wheeled locomotion for payload carrying with modular robot. In *IROS*, 2008.
[2] A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642–653, 2008.
[3] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International conference on Neural Networks*, 1995.
[4] K.Hauser, T. Bretl, K. Harada, and J.-C. Latombe. Using motion primitives in probabilistic sample-based planning for humanoid robots. In *WAFR*, 2006.
[5] T. Krajník, M. Nitsche, J. Faigl, T. Duckett, M. Mejail, and L. Přeučil. External Localization System for Mobile Robotics. In *Proceedings of the International Conference on Advanced Robotics*, Montevideo, 2013. IEEE.
[6] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning, 1998. TR 98-11.
[7] J. Liedke, L. Winkler, and H. Worn. An alternative locomotion unit for mobile modular self-reconfigurable robots based on archimedes screws. In *International Symposium on Mechatronics and its Applications (ISMA)*, 2013.
[8] Paul Moubarak and Pinhas Ben-Tzvi. Modular and reconfigurable mobile robotics. *Robotics and Autonomous Systems*, 60(12):1648–1663, 2012.
[9] K.C. Prevas, C. Ünsal, M.O. Efe, and P.K. Khosla. A hierarchical motion planning strategy for a uniform self-reconfigurable modular robotic system. In *ICRA*, 2002.
[10] R. Primerano, D. Wilkie, and W.C. Regli. A case study in system-level physics-based simulation of a biomimetic robot. *IEEE Transactions on Automation Science and Engineering*, 8(3):664 –671, 2011.
[11] B. Salemi, W.-M. Shen, and P. Will. Hormone-controlled metamorphic robots. In *ICRA*, 2001.
[12] K. Stoy, W.-M. Shen, and P. M. Will. A simple approach to the control of locomotion in self-reconfigurable robots. *Robotics and Autonomous Systems*, 44(3), 2003.
[13] V. Vonásek, K. Košnar, and L. Přeučil. Motion planning of self-reconfigurable modular robots using rapidly exploring random trees. In *TAROS*, 2012.
[14] V. Vonásek, M. Saska, K. Košnar, and L. Přeučil. Global motion planning for modular robots with local motion primitives. In *ICRA*, 2013.