This is an author submitted version of the paper:


@inproceedings{Vonasek13:205389,  
author = "Vonasek, V. and Saska, M. and Kosnar, K. and Preucil, L.",  
title = "Global Motion Planning for Modular Robots with Local Motion Primitives",  
publisher = "IEEE",  
address = "Piscataway",  
year = 2013,  
pages = "--",  
isbn = "978-1-4673-5641-1",  
issn = "1050-4729",  
language = "English"}

For final version, please contact authors at

http://mrs.felk.cvut.cz
Global motion planning for modular robots with local motion primitives

Vojtěch Vonášek, Martin Saska, Karel Košnar and Libor Přeučil

Abstract—The ability to move in complex environments is a key property required for deployment of modular robots in challenging applications like search & rescue missions or space exploration. Wide range of motion types like crawling or walking can be achieved using Central Pattern Generators producing periodic control signals. Although these motions can be very effective to steer robots in their vicinity or in a given direction, they need to be switched to reach a far position in the environment. This paper presents a novel modification of Rapidly Exploring Random Tree (RRT) algorithm for modular robots. For efficient exploration of the configuration space, predefined motion primitives are used. While the motion primitives provide effective local motions, the RRT-based planner switches them in order to reach the desired global goal.

I. INTRODUCTION

Modular robots consist of several modules, which can be connected to form a robot of various shapes. Modules that support autonomous connection/reconnection can be formed into self-reconfigurable robots with possibility to change their structure regarding operational demands. In the comparison to conventional fixed-shaped robots, it may bring additional abilities in applications like space exploration [21], search & rescue missions [2] or object manipulation [9]. Motion planning and navigation are crucial skills to succeed in these challenging applications.

The modular robots utilize two different concepts to move: self-reconfiguration or joint control. In the concept of the self-reconfiguration, the motion is achieved by disconnection and reconnection of individual modules [15], [14].

In this paper, the motion of the robots is achieved by the joint control, where angles of the joints connecting the modules are controlled. To obtain the low-level motions, a concept of Central Pattern Generators is used. The global locomotion is provided by a high-level planner, which is based on the Rapidly Exploring Random Tree algorithm [10].

II. RELATED WORK

Motion planning for modular robots is a challenging task, especially due to many degrees of freedom (DOF). Moreover, the motion of individual modules is constrained by the mechanical connection between them as well as by contacts with the underneath terrain. Such a problem with many kinematic constraints can be solved using sampling-based approaches like Probabilistic Roadmaps [6] or Rapidly Exploring Random Tree (RRT) [10]. The idea of the sampling-based methods is to create a roadmap of the configuration space of robots using a randomized sampling. Then, the required motion in the workspace can be found as a path in the roadmap. The sampling-based planners can cope with dynamic and kinematic constraints. They have been successfully used in many applications including modular robots [20], [22], [19].

The performance of the RRT-based planners depends on the control inputs used to explore the configuration space. Random control inputs can be used for systems with few inputs, but this usually produce non-smooth and clumsy motions in case of modular robots. Instead, predefined motion primitives with known performance and behavior can be used. The motion primitives are basic motion skills such as ‘go-left’ or ‘rotate-right’. The utilization of motion primitives allows to reduce the complexity of the planning task for many-DOF robots like humanoids [8], helicopters [12] or climbing robots [1].

Widely used concept for motion generation are Central Pattern Generators (CPGs), which produce periodic control signal for the actuators. By changing parameters and coupling of the CPGs, various types of gaits like swimming, crawling or caterpillar-like motions can be produced. The CPGs are frequently used for locomotion of modular robots; we refer to [3] for a detailed review. The parameters of the CPGs can be predefined or computed using an optimization technique [4], [5]. The synchronization of control signals produced by CPGs can be based on events generated by neighbor modules [18], or using hormones [17].

A CPG usually provides one type of motion action, like ‘go-forward’ or ‘rotate-left’. While a single motion primitive can be effective for motion in a vicinity of the robot, more motion actions are required to traverse longer distances in complex environments and a switching mechanism between them is required. The main contribution of this paper is the novel algorithm for global motion planning for modular robots. The proposed algorithm employs a set of motion primitives and finds a sequence of them to achieve a global goal position.

III. PRELIMINARIES

In this work, the Scout modules developed in the Symbrion/Replicator projects [11] are employed. The modules are cubes of side length 15 cm and weight ~ 1 kg. They are equipped with two tracks for a fast 2D locomotion. The modules can be connected using four docking mechanisms. One of them is placed on a movable arm, which can move in range \((-\pi/2, \pi/2)\). We refer to [11] for the technical details. An example of a snake robot made of these modules is depicted on Fig. 1a.
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 \( (\alpha, \beta, \gamma) \) 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 \( \mathcal{C}_{\text{free}} \subseteq \mathcal{C} \). The task of the motion planning is to generate a feasible trajectory between an initial configuration \( q_{\text{init}} \in \mathcal{C}_{\text{free}} \) and a goal configuration \( q_{\text{goal}} \in \mathcal{C}_{\text{free}} \).

A pose of a modular robot with \( n \) modules can be achieved by controlling the angles \( a_i \). The motors driving each arm are controlled using a PD controller to achieve the desired angles. A motion of the robot is achieved by changing the angles using a control input signal \( u(t) = (a_1(t), \ldots, a_{n-1}(t)) \).

In this paper, the motion primitives are generated using sinus CPGs: \( a_i(t) = A_i + \sin(2\pi f_i t + \varphi_i) + B_i \), where \( (A_i, B_i, f_i, \varphi_i) \) are parameters of a motion primitive.

For purpose of RRT-based motion planning, it is useful to generate a random control signal. It can be generated using a pure random signal denoted as \( u_r(t) \), where the angles \( a_i(t) \) are generated randomly from range \( (-\pi/2, \pi/2) \). Another approach is to use a randomly initialized sinus CPGs (parameters \( A_i, B_i, f_i \) and \( \varphi_i \) are generated randomly), denoted as \( u_{\text{CPGrand}}(t) \). The difference between these two approaches is examined in Section VI.

### IV. Sampling-based Motion Planning with Motion Primitives: RRT-MP

The Rapidly Exploring Random Tree (RRT) algorithm [10] randomly samples the configuration space \( \mathcal{C} \) of the robot and builds a tree \( T \) of feasible configurations. The tree is rooted in the initial configuration \( q_{\text{init}} \) and it is extended in each iteration as follows. First, a random sample \( q_{\text{rand}} \in \mathcal{C} \) is generated. The nearest node \( q_{\text{near}} \) in the tree is found and expanded to obtain a set of free configurations reachable from \( q_{\text{near}} \). From this set, a configuration \( q_{\text{new}} \) that is nearest to \( q_{\text{rand}} \) is selected and stored into the tree. The algorithm terminates if the goal region defined by the radius \( d_{\text{goal}} \) is approached. The main loop of the RRT algorithm is listed in Alg. 1.

The distance between the configurations is measured using 3D Euclidean metric between the pivot modules, which is sufficient for the global motion planning where small differences between robots’ poses do not need to be considered.

The main contribution of the paper is the RRT-MP algorithm, which is based on the above described RRT algorithm. The contribution of the RRT-MP is the novel usage of motion primitives for exploration of the configuration space. The RRT-MP is described in the following sections.

#### A. Motion model

A motion model of the robot is used in the RRT algorithm to extend the tree towards the random sample \( q_{\text{rand}} \). Usually, an analytic motion model \( \dot{q} = f(q, u) \) is used to obtain a response to a control input \( u \). Due to many constraints caused by interactions between the modules and the terrain, the closed-form motion model cannot be easily derived [16]. As the RRT needs only the resulting configuration, the motion model can be implemented as a black-box using a physical simulation, which is run for time \( \Delta t \) starting from the configuration \( q_{\text{near}} \). Due to randomized selection of nodes for the expansion, the simulated robots need to be placed into different configuration in each iteration of the RRT hence the simulation is not run continuously. To avoid an instability of the simulation, states of the simulated objects including their positions and velocities have to be properly restored before the simulation is run. Each node in the tree thus contains both configuration \( q \) and a vector of simulation variables. We refer to [20] for technical details.

#### B. Using motion primitives for tree expansion

The configuration space is explored by applying several control inputs to the motion model of the robot over time interval \( \Delta t \). In the RRT-MP, these control input signals are used: a) inputs signals defined by motion primitives; b) pure random input signals \( u_r(t) \); and c) signals \( u_{\text{CPGrand}}(t) \) generated by randomly initialized CPGs.

---

**Algorithm 1: RRT algorithm.**

- **Input:** initial configuration \( q_{\text{init}} \), goal configuration \( q_{\text{goal}} \), goal region \( d_{\text{goal}} \), maximum number of iterations \( k \)
- **Output:** path to the goal configuration or failure

```pseudo
T.addNode(qinit); // T is the configuration tree
for iteration = 1...k do

  qrand = generate random sample in \mathcal{C};

  qnear = nearestConfiguration(T, qrand);

  qnew, u = expandConfiguration(qnear, qrand);

  if qnew != NULL then

    T.addNode(qnew, u);

    T.addEdge(qnear, qnew);

    if distance(qnear, qgoal) < dgoal then

      return path in the tree from qinit to qgoal;

  end

end

// trajectory was not found during k iterations;
return failure;
```
Although the global motion of modular robots can be achieved by pure random signals [20], the resulting motions are clumsy and large number of iterations are needed to find a solution. To speed up the planning process, the concept of motion primitives is used in the RRT-MP: instead of generating random inputs, predefined motion primitives with know performance are used. By using the primitives, the motion of the robot during the expansion step is better controlled than with random motions. Moreover, a motion primitive can move the robot over time $\Delta t$ to a more distant location than a random input. Therefore, only few motion primitives are needed to provide the RRT algorithm the exploration ability and consequently, less number of motion model evaluations is needed, which decreases the planning time.

To preserve the probabilistic completeness of the RRT algorithm, both the predefined motion primitives and random signals should 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 $p_r$ in Alg. 2. The number of examined random signals is same as the number of motion primitives (parameter $K$ in Alg. 2).

It has to be ensured, that the motion primitives can be switched into another one. In the case of presented modular robots crawling close to ground, we can assume, that switching of the primitives cannot cause instability of the organism. However, this may not be the case for complex robots like humanoids [8], where the stability needs to be considered during switching of the primitives.

---

**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. **if** rand()> $p_r$ **then**
   3. **foreach** $i$ in motionPrimitives **do**
      4. $\{A_i, f_j, \phi_i, B_j\}$ parameters of $i$-th motion primitive;
      5. $u(t) = (a_1(t), \ldots, a_n(t))$;
      6. $a_i(t) = B_i + A_i \sin(2\pi f_j t + \phi_i)$;
      7. $q = $ physicalSimulation($q_{near}, u(t)$), $t = [0, \Delta t]$;
      8. $S = S \cup \{q, i\}$
   9. **end**
10. **else**
   11. **foreach** ($i = 1..K$) **do**
      12. $u(t) =$ or $u_{CPGrand}()$; // random input
      13. $q = $ physicalSimulation($q_{near}, u(t)$), $t = [0, \Delta t]$;
      14. $S = S \cup \{q, p(t)\}$
   15. **end**
16. **end**
17. $q_{new}, u =$ closest configuration from $S$ to $q_{rand}$;
18. **return** ($q_{new}, u$);

---

**V. GENERATION OF CPG-BASED MOTION PRIMITIVE**

The performance of the RRT-MP is determined by the quality of the used motion primitives. The RRT-MP should be used with such a set of the primitives, that allow to move the robot in the environment. For example, the primitives 'move-step-forward' or 'move-step-back' are sufficient for a robot moving in a tube. However, this set would be insufficient for the same robot moving on a large terrain, because a motion between two random positions in the environment will probably require other primitives such as 'rotate-left'.

For the global motion planning between far locations, the motion primitives should provide motion in all directions. Although it is possible to use many motion primitives (such as 'move-direction-A', where A is an angle of movement), this increases the planning time, because all the primitives need to be examined in each expansion step. We propose to use two types of motion primitives: the move-step-x (where 'x' is forward, back, left and right) and rotate-x (where 'x' is left or right). The effectiveness of these motion primitives is studied in the experimental section.

In this paper, the Particle Swarm Optimization (PSO) [7] technique is utilized to find the CPG-based motion primitives. Standard PSO with 30 particles and 200 generations is used. A particle represents the parameters of CPGs, $p = (A_i, B_i, f_j, \phi_i)$, where $i = 1, \ldots, n - 1$. The parameters are limited to range: $A_i = (0, \pi/2), f_j = (0, 2) Hz, \phi_i = (0, 2\pi)$ and $B_i = (-\pi/2, \pi/2)$. The task of the PSO is to find primitives move-step-x (where x $\in$ {left, forward, right, back}) and two rotating primitives (rotate left and rotate right). The task of the move-step-x primitives is to move robot to the given direction without any rotation. The fitness function for the move-step-x primitives is the distance to a point in the direction of the movement; in the case of rotating primitive the fitness is computed as the angle between starting and final headings of the robots. The fitness is evaluated after 1 s long motion. Examples of found move-step-x primitives are depicted on Fig. 2.

---

**VI. EXPERIMENTS**

The ability of the proposed RRT-MP algorithm to find trajectories between far location in an environment utilizing several motion primitives has been verified in several scenarios. Two modular robots made of Scout modules were used in the experiments: Quadropod and Lizard. The motion

---

**Fig. 2.** Motion primitives of Quadropod (left) and Lizard robots (right). Each primitive is depicted as the trajectory of the pivot module (red curves). The red points represent goals used for fitness evaluation of move-step-x primitives. The robots are placed in their starting positions.
of the robots was modeled using ODE [13] physical engine with 10 ms physical simulation step. The motion primitives described in the previous section were used. The experiments were performed in three environments: Planar with no obstacles (denoted as Plane, Fig. 3), undulating surface with no obstacle (denoted as Surface1, Fig. 6) and undulating surface with a single obstacle (denoted as Surface2, Fig. 7). The dimension of all environments is 23x13 map units and size of one module is 0.5 map unit. The RRT-MP algorithm was terminated if the tree approached the goal region to a distance less than \( d_{\text{goal}} = 1 \) map unit or after \( k = 1500 \) iterations. The time of each edge was set to \( \Delta t = 1 \) s.

The performance of the RRT-MP planner is measured using the number of iterations needed to reach the goal region and by a success ratio. Multiple init/goal configurations were used to evaluate performance of the global motion planning: 126 init/goal pairs were generated uniformly in the tested environment. The measured values are depicted resulting 126 values then represent average performance on environments. For each pair, the RRT-MP algorithm was run using the number of iterations needed to build the tree are shown. However, the run-time is positively correlated with the number of iterations, therefore it is a valuable indicator of the planning speed.

A. Combination of MP and random inputs

In the RRT-MP, the motion primitives are combined with random control inputs. In the first experiment, the influence of the random signals on performance of the planner was investigated. The experiment was performed on the Plane environment. The RRT-MP was run with all four move-step-x primitives with \( p_r \) varying from 0 to 1. For each value of \( p_r \) the motion planning among 126 start/goal pairs was performed, as is described in the previous section. The test was performed for both type of random signals: \( u_s(t) \) and \( u_{\text{CPGrand}}(t) \). The results are shown on Fig. 5.

The solutions were found in most cases when only the move-step-x primitives were used (\( p_r = 0 \)). In this case, the primitives of Lizard were more effective (the success ratio is higher than 90 %) than for Quadropod, where the success ratio varies from 0 % to 100 %. Therefore, the Lizard was able to traverse each start/goal pair with 90 % success ratio, while the Quadropod failed on several start/goal pairs. As the \( p_r \) increases, the number of iterations (the top graphs on Fig.5) increases for both types of random signals. The influence of increasing \( p_r \) to success ratio differs for both robots: while the ratio decreases for Lizard, it increases for Quadropod.

The opposite effect of \( p_r \) for the robots is caused by different behaviors of the primitives: the move-step-x primitives of Quadropod are more straight than the primitives of Lizard. For example, during the forward motion along 'move-forward' primitive, the Quadropod does not change its heading as much as the Lizard. The change of robot's heading (error) for the move-step-x primitives can be seen in Tab. I. The Lizard rotates especially during execution of 'move-right' and 'move-left' primitives. This is caused by the distance-based fitness function which does not consider
TABLE I
HEADING ERROR (IN RADIANS) OF THE ‘MOVE-X’ PRIMITIVES.

<table>
<thead>
<tr>
<th>Robot</th>
<th>forward</th>
<th>left</th>
<th>right</th>
<th>back</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lizard</td>
<td>0.04</td>
<td>0.21</td>
<td>0.48</td>
<td>0.03</td>
</tr>
<tr>
<td>Quadropod</td>
<td>0.03</td>
<td>0.01</td>
<td>0.02</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Fig. 6. Examples of found solutions for Quadropod robot on the Surface$_1$ environment with motion primitives (top) and random inputs (bottom).

The random control signals usually lead to a short motion but with a rotation. Therefore, by increasing the probability of using the random signals, the Quadropod performs more rotations. This effect is achieved naturally by Lizard’s move-step-x primitives.

For low $p_r$, the performance between RRT-MP with random signals $u_r(t)$ or $u_CPGrand(t)$ is not significant and we can conclude that both types of random signals can be used.

The graphs on Fig. 5 also show, that for $p_r = 0.95$, i.e. when the motion primitives are used only in 5% of iterations, the RRT-MP is able to find a solution with low success ratio. The performance can be improved by allowing more iterations in the RRT-MP. However, the results suggest, that even pure random signals could be used in the planner, which was presented in our previous work [20]. The examples of found trajectories found with $p_r = 0$ and $p_r = 1$ are depicted on Fig. 3, 4, 6 and 7.

B. Influence of rotation primitives

In the second scenario, the influence of motion primitives to the performance of RRT-MP planner is studied. Here, only the motion primitives without random control signals are used ($p_r = 0$). The experiments were performed with four sets of motion primitives: a) move-step-x motion primitives (M); b) move-step-x and rotate-left and rotate-right (ALL); c) move-step-x and rotate-left (M+RL); and d) move-step-x and rotate-right (M+RR). The usage of only rotating primitives was not tested, because they do not move robot far enough and reaching a goal would be time consuming. The performance of the RRT-MP with these sets of motion primitives is depicted on Fig. 8 and 9.

On the Plane environment, the move-step-x Lizard’s primitives provide enough skills (Fig. 8 left column) to move a robot between random start/goal locations. This is indicated by high success ratio, which is 100% for almost all start/goal pairs. The performance is little increased on Surface$_1$ and Surface$_2$ environments when rotating primitives are used, however it is not statistically significant. We can conclude, that involvement of the rotating primitives does not help the Lizard to move on the tested environments.

However, the usage of the rotating primitives have significant impact on the Quadropod robot. On the Plane environment (Fig. 9 left), the usage of only move-step-x primitives lead to low success ratio (the success ratio varies from 0% to 100%). This is also indicated by high number of iterations. The performance is significantly improved if the two rotating primitives are employed (column ALL on Fig. 9). In this case, the ratio is 100% and the solution is found in less than 100 iterations.

The usage of the rotating primitives does not improve the performance on the Surface$_1$ and Surface$_2$. This is caused by unwanted rotation of the move-step-x primitives. The change in the heading when moving by the move-step-x primitives is larger on Surface$_1$ and Surface$_2$ than on Plane. Therefore, involvement of the rotating primitives improved the performance on the Plane environment for Quadropod. It did not improve the performance for the Lizard, because the Lizard has implicit (error) rotation in its primitives.

The experiments have shown the ability of the RRT-MP to
find motion between far locations on the tested environments.
A trajectory was found even if only a small set of the primitives was used. The importance of the richness of the utilized primitives has been shown by both experiments: to move between arbitrary positions in the environment, both move-step-x and rotating primitives are required.

VII. CONCLUSION

A novel motion planning technique for the modular robots has been presented. The proposed RRT-MP planner utilizes predefined motion primitives, which are switched by the RRT-based planner to achieve a global goal. The experiments have shown that the RRT-MP is able to solve the planning problem between distinct locations. For efficient movement in the environment, a set of rich motion primitives should be used. Both experiments have shown, that pure move-step-x primitives are not rich enough for the purpose of motion planning among arbitrary locations. However, if also rotation primitives are employed, the RRT-MP algorithm is able to find solution of the global motion planning.

VIII. ACKNOWLEDGMENTS

The work of V. Vonasek and L. Preucil was supported by the Ministry of Education of the Czech Republic under projects 7E11017, by the EU under the project ICT-216342 and by TACR grant No. TE01020197. The work of K. Kosnar by TACR grant No. TE01020197 and by GACR grant No. 13-30155P. The work of L. Preucil was supported also by CTU grant No. SGS10/145. The work of M.aska was supported by GACR under grant no. GP0103/12/P756. The experiments have been run using Grid Infrastructure Metacentrum (project No. LM2010005).

REFERENCES