Roombots—Towards Decentralized Reconfiguration with Self-Reconfiguring Modular Robotic Metamodules

Alexander Sproewitz[‡], Philippe Laprade[‡], Stéphane Bonardi[‡], Mikaël Mayer[‡],
Rico Moeckel[‡], Pierre-André Mudry[#] and Auke Jan Ijspeert[‡]
EPFL Ecole Polytechnique Fédérale de Lausanne, Switzerland

***Abstract*— This paper presents our work towards a decentralized reconfiguration strategy for self-reconfiguring modular robots, assembling furniture-like structures from Roombots (RB) metamodules. We explore how reconfiguration by locomotion from a configuration A to a configuration B can be controlled in a distributed fashion. This is done using Roombots *metamodules*—two Roombots modules connected serially—that use broadcast signals, lookup tables of their movement space, assumptions about their neighborhood, and connections to a structured surface to collectively build desired structures without the need of a centralized planner.**

I. Introduction

Self-reconfiguring modular robots (SRMR) are modular robots (MR) extended by an active connection mechanism (ACM). The ability to autonomously attach modular robot units with each other enables the creation of almost arbitrary robot structures. Single modular robotic units are designed with a low degree of freedom, usually between one and three. This restricts a single modular robot in its locomotion and reconfiguration abilities. To overcome an obstacle or to manipulate an object the modular robot collective is needed. It is the idea of SRMR systems that units can attach with each other task-dependently, i.e. a goal structure is chosen and executed that is most suitable for a given situation. This criteria might for example be the resulting robot shape, type and number of degrees of freedom, joint torque limit, or the overall number of modules. We are developing a homogeneous, self-reconfiguring modular robot system named Roombots (RB). Roombots are designed as building blocks for furniture that moves, self-assembles, self-reconfigures, and self-repairs. Reconfiguring RB modules into furniture-like structures can be described as a sub-problem of general modular robot reconfiguration. Several approaches for centralized or distributed reconfiguration have been proposed so far. Depending on the level of abstraction and the assumptions made, those methods partially or even completely solve modular robot reconfiguration. It is useful to design a general reconfiguration strategy such that it can be applied to a number of modular robot systems with different characteristics. Usually a layer of abstraction is needed. One very intuitive example is the "sliding cube" model, which uses simple translational motions of individual modules and is the basis for a number reconfiguration strategies (discussed in more detail in Section II). If the derived strategy using the above simplified representation can be implemented on a hardware modular robot system, the task of reconfiguration is actually solved. However often assumptions concerning collision between modules, asynchronous movement, communication, sensor range and type, or consensus based decision making are violated with the transfer to the real robot system. In this work we design a largely decentralized reconfiguration strategy which guides RB metamodules during a reconfiguration sequence into an adaptive structure, e.g. a stool or a chair or RB modules. RB metamodules use an inch-worm type of locomotion in a structured environment (i.e. the floor has passive connectors embedded) for reconfiguration. This work does not invent a new strategy but rather uses a number of existing strategies not only from the modular robot community. We are aiming at a largely distributed method (for scalability reasons) which takes into account the kinematic constraints of the Roombots modules, as well as restricted sensing, communication and computational aspects. We are addressing the three following questions: (i) Four possible metamodule configurations exist (Fig. 3). We are interested whether one of them shows better reconfiguration performance. (ii) We are applying three different strategies to force-field guide the reconfiguration of RB metamodules. In other words we want to know how much repelling force between metamodules is needed/optimal to avoid collision while moving towards a goal position. (iii)

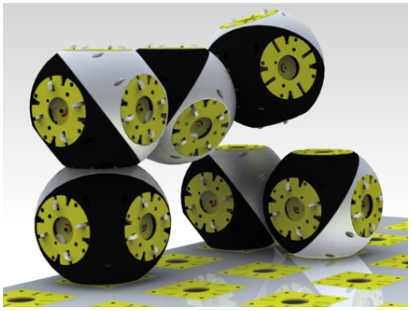


Fig. 1: Rendered vision of the Roombots project: a Roombots metamodule (left, two Roombots module connected serially) is attached at its foot hemisphere (black segment, very left) to the ground, and is in the process of coupling to a single Roombots module (back). The ground is "structured", i.e. passive connectors with the same patterns as Roombots' active connectors are embedded in the environment.

[‡]Biorobotics Laboratory at EPFL, CH-1015 Lausanne, Switzerland, alexander.sproewitz@epfl.ch
[#]Systems engineering, University of applied sciences western Switzerland CH-1950 Sion, Switzerland, pandre.mudry@hevs.ch

In our current framework a human operator (ideally a lay user) is providing the blueprint for the structure to be built (e.g. a chair). We would like to see the influence of the building order, that is which positions of a given structure should be filled first by metamodules to avoid dead-locks and collisions.

The paper is structured as follows. In Section II we describe shortly the concept of modular robots, and the most common reconfiguration strategies applied to modular robots. We present the Roombots hardware in Section III. We explain our reconfiguration strategy in detail (Section IV), results are discussed in Section V. A conclusion sums up the work (Section VII).

II. Related Work

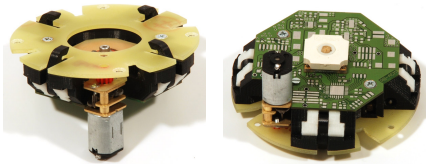
With the Roombots project we wish to extend but also test a future scenario, where technology is being merged into every day environment, ranging from tables to walls, from furniture like shelves to tangible or interactive "roomware" [1]. Ultimately adaptive Roombots furniture will be able to transform and merge from one shape, e.g. two chairs into another, e.g. a table. We use the concept and the ideas of *self-configuration modular robots* (SRMR) or *Dynamically Reconfigurable Robotic Systems* [2] as physical building blocks for our adaptive furniture. The field of self-reconfiguring modular robots, which are modular robot units that can actively attach and detach themselves with each other and the environment, is a more recent robotic concept which was firstly implemented with CEBOT ("cell structured robot") [2] in the late 90's. Depending on the capabilities of a single modular unit, almost arbitrary shapes can be created by remote control [3]. This is especially helpful if the task is initially unknown. If a quadruped-shaped modular robot locates a hole in the wall it can change shape into a caterpillar-like structure, and go through. As many different robotic shapes can be created with the same set of units, transport is easy and less costly, e.g. to remote locations. Units are interchangeable such that modular robotic cells can be replaced in case of failure, what potentially makes these systems robust. However these advantages come with a price. Implementing autonomy in modular robots, equipping each of the units with a connection mechanisms, actuators, and electronics makes them heavy, expensive, and hard to design. A robotic configuration built from modular robots will normally perform less well compared to a monolithic robot as the abilities and dynamics of a monolithic robot can be optimized—it serves a smaller number of dedicated, pre-known tasks.

The usefulness of a modular or monolithic approach therefore depends on the application. Research in modular robots aims towards applications at disaster sites, remote or hazardous environments, where their shape changing characteristics and robustness are crucial. A number of modular robot projects are working at micro-scale modular robots, i.e. they aim for rapid prototyping-like technologies [4]. For the Roombots project we chose self-reconfiguring modular robots for their abilities in building arbitrary, adaptive furniture.

Finding and applying an automated controller to change shape is one of the main topics in reconfigurable robotics, where decentralized strategies outweigh centralized strategies. The latter often use a graph-based approach, describing the combined modular robot structure using graph theory, where actions are represented by insertion and deletion of edges and vertices [5]. Connector actions and joint rotations are the result of an optimization process attempting to morph the graph representing the initial structure, into the goal configuration. This allows for a very precise reconfiguration process, however graph methods do not scale well with increasing numbers of joints, connectors and modules. Common approaches for decentralized reconfiguration are "cluster flow" [6] locomotion or "water flow-like locomotion algorithms" [7] and describe locomotion by self-reconfiguration (or vice versa). They facilitate large numbers of, usually abstracted modular units moving or changing shape through the environment, where units are simulated as a cubes or spheres which slide along planes and around edges, or rotate around edges [8]. Movements of single units can be guided by a global gradient [9] or triggered by hormone-like messages [10]. Cellular automata [11] oriented methods use distributed, reinforcement learning algorithms to optimize the behaviour of single units task dependent, and with only partial world-knowledge [12]. Such strategies enable enormously scalable systems [13]. Using a simplified modular robot unit presentation, like the above "sliding cube" model is helpful to derive a reconfiguration strategy on an abstracted level. To implement the strategy on a low-level, i.e. on an actual modular robotic systems, the notion of *metamodules* is formulated. Metamodules are local, clustered assemblies of modular robot units which are combined for the purpose of moving just as their sliding-model counterpart cubes, however by using the actual degrees of freedom available from the hardware units. Butler and colleagues [7, cf. page 7] mention the usefulness of such metamodules (Molecule's *tile* [14] and Atom's *grain* [15]). Dewey and colleagues [16] cluster the entire modular robot assembly in equal, non-dense generalized metamodules, which enables them to apply a very simple planner for module movement through the structure.

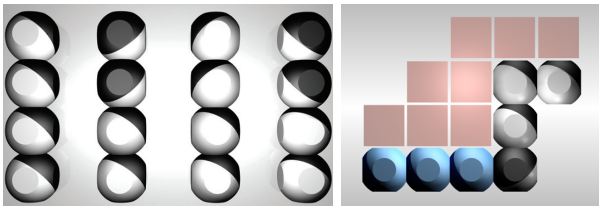
Roombots are similar in their degrees of freedom (DOFs) to the *3D Molecubes* [17], and have inherited some of their main movement characteristics. Roombots feature one additional DOF, and we combine two Roombots (RB) modules serially into one RB metamodule. We are interested in building furniture-shaped structures with metamodules in the centimeter-scale, hence we can settle with medium-large number of modules. Also we can make use of connectors embedded in the environment and broadcast communication for our application, and are able to omit some of the hard constraints such as constant connectivity, and local communication. We are still interested in a distributed system with low demands on communication bandwidth. A strong constraint of our Roombots system is the movement space of

a single Roombots module—six DOF connected serially are very powerful, in terms of being able to overcome concave or convex obstacle edges. However a Roombots metamodule requires a rather large space around itself to move which needs to be considered in advance of the movement.



(a) Outside plate ACM. (b) PCB plate ACM.

Fig. 2: Active connection mechanism (ACM) of the RB. Four mechanical latching fingers grab synchronously into the neighbouring module or the structured surface. The mechanism is actuated with a mini-DC motor, position of the grippers is sensed with a potentiometer (Fig. 2b at the center). The ACMs are designed to be mechanically autonomous, any other type of connector could be plugged into the corresponding RB sockets.



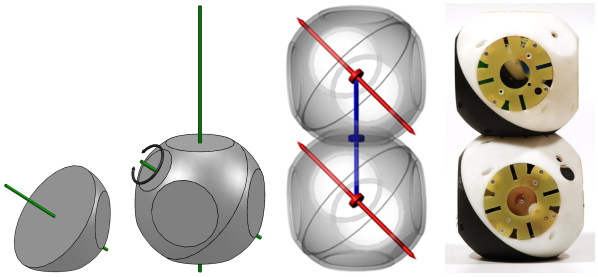
(a) (b)

Fig. 3: (a) Four different possibilities for RB metamodules exist by (semi) locking two RB modules serially together. We use the relative orientation of the center axis of the two ACM-connected hemispheres for naming: (from left to right) *parallel PAR*, *perpendicular PER*, *shear-S SRS* and *shear-Z SRZ*. The orientation of the lower RB unit is kept fixed in all pictures. (b) *Shape-transition* of a metamodule, from I-shape configuration (bluish, horizontal) to L-shape. Red boxes indicate the *collision cloud* a metamodule transition is producing, where every touched cube in the 3D grid is being recorded. RB movements are in 3D, this figure shows only a frontal projection of the cloud.

III. Hardware

Similar to other modular robotic systems Roombots (RB) units [18] are fitted into a regular cubic grid. We are using a grid size with 110 mm edge length. We connect two RB modules serially into a *RB metamodule* (Fig. 3), four combinations are possible. Each resulting metamodule has its own range of motion and movement characteristics. Any of the three joints (Fig. 4c) of an RB unit delivers sufficient torque to rotate a metamodule in the "worst case scenario situation", i.e. out of a horizontal stretched position. RB modules are fabricated mostly from 3D printed ABS plastics pieces, plate-elements are milled out of glass-fibre sheet material. An RB module weights about 1.4 kg, that includes battery power for an estimated 30 min of continuous actuation, and the weight for electronic boards[1]. Joints are equipped with high gear ratio gearboxes ($\sim 366:1$ and $\sim 305:1$), actuated by strong DC motors which results in 5 Nm and 7 Nm torques for middle and outer joints, respectively. Any of the three joints is continuously rotational, i.e. can turn without mechanical stop. Electrical power and communication are transmitted with slip rings within the unit. The two outer DOFs of an RB unit (Fig. 4c, red) are of the same type as in the Molecube modules [17],[19]. RB units have an additional actuated swivel joint (Fig. 4c, blue) in-between. The high torque demands and the resulting high gearbox ratio values limit RB' maximum rotational speed, the center joint needs 3 sec to rotate 360°, both outer joints roughly 2 sec. RB's active connection mechanism (ACM) is genderless, four-way symmetric, with four mechanical latching fingers (Fig. 2a) which are completely retractable inside the body. Connector units are roughly 65 mm in diameter and fit into any of ten dedicated sockets of an RB unit. In many ways the connector design is similar to the AMAS connection mechanism [20], although we use a different trajectory for the movement of the latching fingers [21]. Initial connector tests indicate a passive tolerance against alignment errors of roughly 2 mm between modules, we have also good first results for detaching under load. We are in the process of finishing the RB hardware, hence all the experiments are implemented in Webots [22], a physics-based simulation environment. The simulation takes available joint torques, velocity limits, weight, the geometry including active connection mechanisms, axes, and hemispheres shapes into account.

[1]The electronic hardware for RB is under development.



(a) (b) (c) (d)

Fig. 4: An RB module is made of four (a) hemispheres. (b) Half a RB metamodule has the same dof as a *Molecube* [19], two such spheres are connected with a swiveling joint into a full RB module (c) with three DOF. All joints are continuously rotational, that is they turn without a mechanical stop. (d) shows the picture of a hardware RB unit. The reconfiguration method proposed in this work requires only two ACMs, always one is located in the most outer hemisphere of a RB module. The remaining 8 sockets of the RB module are filled with passive connector plates.

IV. Distributed Reconfiguration

This section describes our initial, currently simulated, approach to reconfiguration by locomotion on a structured surface, i.e. in a 3D environment with embedded connectors to which units can attach. RB metamodules are the moving units, and their movements are guided towards the next active seeding position by a force field. Metamodules send and receive broadcasts among each other to gather knowledge of their nearest neighbourhood. A set of shape-transitions and corresponding collision-clouds (Fig. 3b) stored in a look-up

table enables each metamodule to largely avoid collision, with itself, other metamodules and the environment. We finish the section with initial results characterizing Roombots (RB) metamodules for this type of reconfiguration by locomotion.

A. Strategy

We will explain the distributed reconfiguration mechanism with RB metamodules on the example of building a chair-like structure, e.g. Fig. 7.

a) Metamodule initialization: RB metamodules are initially placed in our structured environment (see the connectors in Fig. 1). A metamodule starts by being attached to a connector with its foot hemisphere. It then determines its initial position and orientation (on the real units this will be done by local communication with the connector or reading out a tag on the connector's surface). RB also have the ability to sense their own shape, by reading out internal joint angle sensor values.

b) Seeding recipe and metamodule shapes: The metamodule now receives information about its environment, e.g. obstacles or walls, but most importantly the *seeding recipe* of the goal structure. The seeding recipe is the, currently hand-coded, "blueprint" for the structure which will be assembled from all the metamodules around, e.g. a chair-like structure (Fig. 7). It will be provided by a human operator. The seeding recipe includes the position and the order of the *seeding cubes*, which are attachment points for a metamodule within the goal structure. Metamodules are not assigned to a specific seeding cube, but the first arriving metamodule will fill the active position, and send a broadcast telling the seeding cube is taken. Remaining metamodules will switch and go towards the next seeding cube in the seeding recipe. The recipe also includes the information of what type of metamodule-shapes the structure will be built from. Metamodules can take five possible shapes: I, L, S, U and $3D-S$ (please see [18] for details). Fig. 3b shows an I-shaped metamodule being rotated into an L-shaped metamodule, Fig. 8 an L-to-L shape transition, and a L-to-3D-S shape transition highlighted.

c) Messages and locomotion: Metamodules use shape-to-shape transition for a slinky-toy like walking in 3D. Before a shape-transition, a metamodule sends a broadcast status message which contains its foot position and its ID. The broadcast messaging is meant as a replacement for close-range sensing of other metamodules, and serves to avoid colliding with them. This requires the knowledge of absolute coordinate points for all metamodules and the goal shape, which is possible in our semi-large environment.[2] A module can derive its neighbourhood from those status messages by comparing the senders position against its own. It will store this information for one step, and only for modules in close range.

d) Force-field guidance: The metamodule now knows its own absolute position $\vec{D}_{foot} = [D_x D_y D_z]$ in the 3D grid, the position of k number of current seeds $\vec{D}_{seed}$, and the positions $\vec{D}_{meta}$ of n number of neighboring metamodules in range. It calculates a force vector $\vec{V}_f$ by summing up the distance vector from the active seeds (attracting "sinks"). Depending on the strategy, neighboring metamodules are included in this calculation. They represent "sources" and emit a repelling force field, with a negative sign. At last the metamodule reaches for the next closest connector in the direction of $\vec{V}_f$. Once the metamodule head is connected to its new position, the module unlocks the foot, sends a new status message and repeats the cycle.

$$\begin{aligned} \vec{V}_f &= \sum_{i=1}^{k} \frac{\vec{D}_{foot} - \vec{D}_{seed_i}}{|\vec{D}_{foot} - \vec{D}_{seed_i}|} \\ &- \sum_{j=1}^{n} \alpha(\vec{D}_{foot}, \vec{D}_{meta_j}) \frac{\vec{D}_{foot} - \vec{D}_{meta_j}}{|\vec{D}_{foot} - \vec{D}_{meta_j}|} \end{aligned} \quad (1)$$

$$\begin{cases} \alpha(\vec{D}_{foot}, \vec{D}_{meta_j}) = 0 & (a) \\ \alpha(\vec{D}_{foot}, \vec{D}_{meta_j}) = \frac{1}{4}(|\vec{D}_{foot} - \vec{D}_{meta_j}| - 4) & (b) \\ \alpha(\vec{D}_{foot}, \vec{D}_{meta_j}) = 1 & (c) \end{cases} \quad (2)$$

e) Force vector strategies: We are interested in different strategies concerning the influence of neighboring metamodules at the $\vec{V}_f$ calculation, and have designed three modes which are switched with the α function: (a) The α-greedy approach ($\alpha = 0$), where neighboring modules have no influence on the force field of other metamodules. During reconfiguration metamodules should go as straight as possible towards the next active seeding position. To minimize collision, modules pause as soon as they detect (by comparing broadcast messages) another metamodule in a very close range, i.e. within four cubes distance. The lock is released with the next status message. (b) A α-slope function, where $\alpha = \frac{1}{4}(|\vec{D}_{foot} - \vec{D}_{meta_j}| - 4)$. This gradually decreases the repelling force between four and eight cubes distance. (c) A α-step function, where $\alpha = 1$. Any metamodule within eight cubes distance provides a full force component. The hypothesis guiding this experiment is that with an additional, repelling force component metamodules will have a tendency to keep a minimum distance between each other. Hence less collisions should occur.

f) Look-up table and collision-cloud computation: As we do not apply sensing in the conventional sense, there is the danger of collision within a metamodule, between metamodules, or with an object. We have designed a method that in-advance calculates what we call a *collision cloud* (Fig. 3b) of a single metamodule for all permutations of initial and final metamodule shapes.[3] The collision cloud represents the number and position of the virtual cubes being touched during the transformation, and is stored in a lookup

[2]Implementing reliable proximity sensing at the hardware level is very complicated, global communication will exist also for other purposes, and can be re-used here.

[3]There are five possible metamodule shapes, and four different metamodule configurations. Each can be assembled with different joint values. Three positions are possible for each of the four outer RB DOF in a metamodule, and four positions for the two inner DOF.

table in an external device. At the begin of each step the metamodule will request the collision cloud corresponding to its initial and final shape from the look-up table. It then checks, based on the cubic grid, if the cloud intersects with any known object or metamodule in range. The look-up table enables us to centrally store data which would be hard to compute in real-time for a single module, and is repeatedly requested from many metamodules.

V. Results

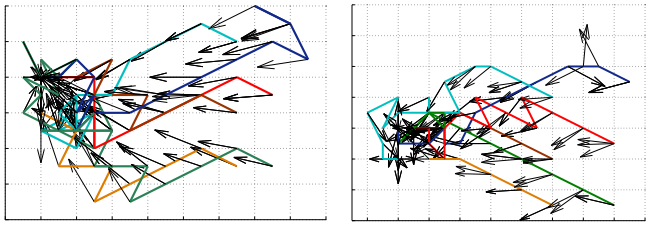
We performed experiments on two simple furniture-like structures: (I) a non-dense cube-like structure built from four metamodules, and (II) a chair-like structure built of six metamodules. In both experiments all metamodules where initially placed about 20 steps Manhattan-distance away from the goal structure. The seeding plan of the goal structure was made by hand. Each cube-setup was repeated three times with shifted initial conditions, i.e. the starting points of the metamodules are moved randomly by one or two fields. For the chair-setup we altered the seeding recipe of the chair structure; (i) in the first run chair-leg seeds are given in a circular order, seed number four and five are metamodules for the back of the chair. (ii) the four legs of the chair are given in a cross-wise order, then the seeds for the back of the chair. (iii) all chair-legs have the same seeding priority, again the seeds for the metamodules on the chair's back are given last (iv) has the same conditions as (ii), however initial positions of the six metamodules are shifted randomly by up to two cubes. All experiments were tested with four different types of metamodules (PAR, PER, SRS, and SRZ, see Fig. 3), and three different reconfiguration strategies (α-greedy, α-slope, α-step function).

TABLE I: Four metamodule cube assembly: numbers show collisions (CL, numerical values), dead-locks (DL) are indicated by $*$. Table rows indicate three different strategies: α-greedy, α-slope and α-step function for the force vector estimation. Columns show the four different meta-module configurations. Three sets of experiments per configuration are shown, with the initial position of the meta-modules randomly shifted by a small number of Manhattan distance steps. Numerical values in parenthesis include the number of collisions in dead lock cases. We exclude dead-lock cases to counting collisions, as those values are not too meaningful.

	α-greedy	α-slope	α-step	DL	CL
PAR	4∗ 0 0∗	0∗ 2 12∗	0∗ 2 2∗	6	4 (22)
PER	5∗ 0 2∗	1∗ 1 0	0 1 0	3	2 (10)
SRS	0 0 0	0∗ 3 5	0 0 1	1	9 (9)
SRZ	0 1 0	0 0 0	1 0 0	0	2 (2)
DL	4	4	2	10	
CL	1 (12)	4 (24)	5 (7)		

a) Results experiment I, 4 metamodule cube: Fig. 5a and Table I show the results for this experiment[4]. All three strategies were tested, with four metamodule configurations, and three random initial conditions ($3 \times 4 \times 3 = 36$ experiments). In 26 of 36 experiments the final configuration was reached and the shape was created, in 17 cases without collision between metamodules. In all cases the area around the final configuration was reached. Modules needed about 9 to 45 moves to reach their targets and presume their seeding postures.

[4]Additional movies can be found at the Roombots page *http://biorob.epfl.ch/page38279.html*



(a) (b)

Fig. 6: Top view onto the trace-patterns of six metamodules (depicted with different colors) moving towards their seeding points in the left center area. Trace points show the pivot point of the foot-hemisphere of each metamodule. Quiver plots show the direction of attraction at iterative steps. Left figure: due to the α-greedy strategy PAR metamodules aim directly for their next seeding position. They will only be paused by a close-by metamodule with a higher priority. Right figure: the same experimental setup, but with SRZ metamodules, and a strategy applying the α-slope function (2). Quiver orientations in (b) indicate that metamodules are being repelled among each other on their way to the seeding position.

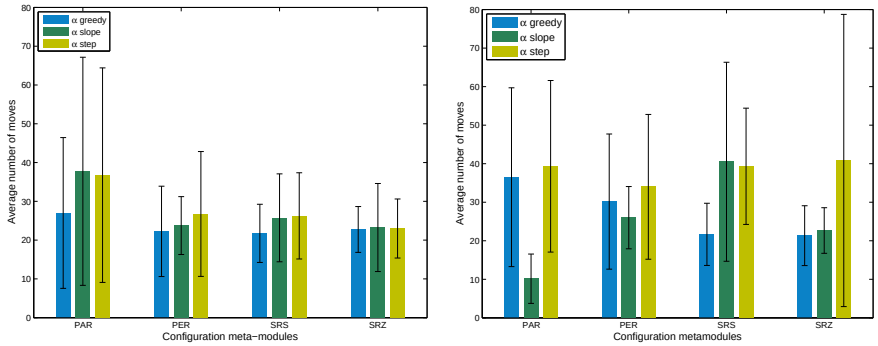
TABLE II: Result table showing dead-locks (DL and $*$) and collisions (CL, numerical values) for the reconfiguration into a 6 metamodule chair-like structure. Four experiments for each combination of (PAR, PER, SRS, SRZ) and (α-greedy, α-slope, α-step) are performed altering mainly the *seeding recipe*. Meaning of numerical values is the same as in Table I.

	α-greedy	α-slope	α-step	DL	CL
PAR	0∗ 0∗ 1∗ 3	7∗ 1∗ 1∗ 0	9∗ 2∗ 1∗ 2∗	10	3 (27)
PER	1 6 0∗ 8	2 1 2 1	5 1 1 1∗	2	28 (29)
SRS	0∗ 2 1 0	1∗ 1∗ 1 0∗	8 7 0 6	4	25 (27)
SRZ	0 0 2 0∗	0∗ 2 0 3	2 0∗ 1∗ 4∗	5	9 (14)
DL	6	7	8	21	
CL	25 (29)	12 (23)	30 (50)		

b) Results experiment II, 6 metamodule chair: Fig. 5b and Table II show the results for this experiment, Fig. 6 shows tracing patterns of metamodules for one experiment with a α-greedy and a α-slope-strategy, respectively. Fig. 7 show a snapshots series of a successful α-slope run with SRZ metamodules. All three strategies were tested against all four metamodule configurations. The seeding recipe was altered (see above; $3 \times 4 \times 4 = 48$ experiments). In 27 of 48 experiments the final configuration was reached and the chair was created, in 6 cases without collision between meta-modules. Modules needed approximately 15 to 55 moves to reach their targets and presume their seeding postures.

VI. Discussion and Future Work

Both experimental setups included successful trials, for several combinations of force field strategies and meta-module configurations. This shows that already by straightforward hand-coding a seeding recipe, structures can be built with the presented force-field guided strategy. We conclude further that at least two metamodule configurations (PER and SRZ) together with the greedy or slope reconfiguration strategy appear to be the most promising method for distributed



(a) 4 metamodules cube (b) 6 metamodules chair

Fig. 5: Bar plot showing the average number of moves four/six metamodules (PAR, PER, SRS, SRZ type) need to built a cube/chair-like structure, respectively. Each bar represents one experiment. Colors indicate different force-field strategies (α-greedy, α-slope, α-step function). (a) Building a cube from four metamodules was the easier task of both, about 20 moves are needed in average. PAR metamodules performed worst in average, otherwise plots for the SRZ metamodules indicate a small superiority. As a tendency applying the α-greedy strategy results in the least amount of necessary moves. (b) The chair structure is more complex, the results show more clearly now that the α-step strategy needs longer to assemble. Only PAR and SRS metamodules succeed to complete the chair with this strategy. The α-slope strategy performs very well in both successful cases (for PER and SRZ metamodules).

reconfiguration. In the following we discuss a number of options to avoid collisions between modules, and dead-locks in future.

a) Mixing metamodules: In this work we concentrated on testing metamodules in their four possible configurations (PAR, PER, SRS and SRZ), however we never make use of the reconfiguration ability *within* a metamodule. In case it is possible to configure into e.g. a metamodule U-shape on the ground (that is if a connector is reachable for this configuration), any metamodule can switch its configuration in runtime. This should increase possible shape transitions, as for example a PER metamodule could change into a SRZ metamodule, if the lookup table indicates better solutions.

b) Seeding recipe: Results indicate that metamodules get stuck mostly within the last sequences of the reconfiguration. Furthermore experiments with slightly altered seeding recipe (6 metamodule chair) show that optimizing the seeding recipe will improve reconfiguration, and in more cases the final configuration can be reached.[5] Building a good seeding recipe presents a non-trivial task; Roombots metamodules have a rather complex movement characteristics, which influences not only their ability to move over a structure, but also how this structure can be assembled. Especially the orientation of the foot hemisphere plays a large role, however this is not included yet in this work. We are currently looking for a scalable planner to automatically come up with a seeding recipe based on a CAD presentation of a structure, to help a lay user with designing furniture-shaped structures.

c) Asynchronous vs. cyclic: The presented framework applies the RB metamodules in an asynchronous manner, nothing is coordinated. Even if all metamodules start at the same time, and as different joint angles need to be reached, metamodules will de-synchronize rapidly. Missing sensing abilities, and asynchronous metamodules assuming neighborhood knowledge based on communication is the reason for collisions; A metamodule sends a status message, checks its environment, finds it unoccupied and starts to move. If another close-by metamodule moves with a delay, it assumes neighborhood knowledge on an outdated basis, and resumes movement in the shared space of another metamodule. There are at least two solutions. (i) An immediate solution to completely avoid collisions and largely avoid dead-locks features a single active metamodules at all time, with the other metamodules being paused at their starting positions. However reconfiguration time to completion will increase largely. (ii) One could increase the safety distance between metamodules, e.g. to ten cubes. It is then physically impossible for two moving metamodules to meet within one step, assuming that both move with about the same speed. However this requires large distances between metamodules. (iii) Another option could be consensus-based decision making between metamodules, to agree on one's priority. This could benefit from a global clock, i.e. synchronized cycles of movements as described in [23]. Once a synchronization would be implemented, the overall setup becomes deterministic, at any crucial point in time (beginning of cycles) distances between metamodules are known, communicated, and a consensus could be made. However this strategy will require additional hand tuning to find proper safety distances, and might also involve the creation of sets of rules to avoid e.g. blocked Roombots locking each other.

VII. Conclusion

In this paper we have presented a decentralized approach to reconfiguration with Roombots metamodules. Reconfiguration through locomotion uses Roombots metamodules applying slinky-toy like movements attaching at embedded connectors in the environment to move and change shape. Metamodules are attracted and guided by a virtual force-field, they use broadcast signals, look-up tables of collision clouds and simple assumptions about their near environment to reach their seeding positions, which are currently hand

[5]Indicated by the amounts of dead-locks for each chair-experiment type: 6, 5, 5 and 5, the first experiment had a less good seeding recipe.

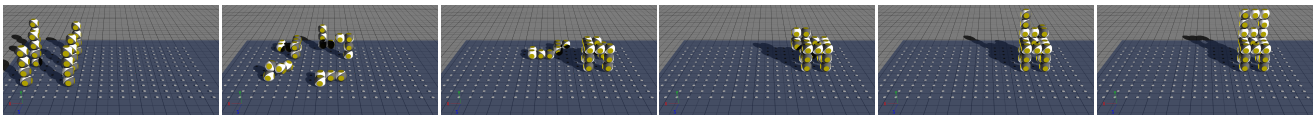


Fig. 7: Snapshots series of six SRZ Roombots metamodules reconfiguring into a chair-like structure, from left to right. The applied force field strategy is α-slope based, the reconfiguration sequence and the order of pictures is from left to left to right. Metamodules start at in a straight posture (left side). They attach and detach at passive connectors embedded in the ground, and use them as pivot points for a slinky-toy-like motion. Once a metamodule reaches a goal point within the chair-structure, it switches off. Newly approaching metamodules will eventually move over it.

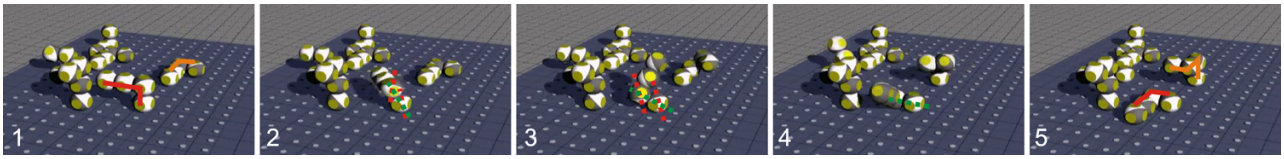


Fig. 8: In this snapshots series two shape transitions are highlighted. (1) shows an L-shaped metamodule in the front (indicted with red frame), and a L-shaped module in the back (orange frame). Both modules transform during the five snapshots. The front metamodule uses three of its lower DOF to transform in another L-shape. The hind metamodule changes into a 3D-S shape. For both modules this presents one step, a series of this steps can be as a slinky-toy like movement, where head and foot module are alternated.

coded. We presented results from simulation tests with two structures (a non-dense cube and a chair) made of four and six metamodules, respectively. Four different metamodule configurations and three force-field models were tested, we also investigated the influence of altering the seeding order of the goal structure. Future research on reconfiguration will additionally explore how to include passive elements into the reconfiguration. We are aiming at testing and using more advanced seeding recipes, runtime metamodule changes, and reconfiguration based on cyclic movements.

VIII. Acknowledgement

This project has received funding from the EPFL and from the European Community's Seventh Framework Programme FP7/2007-2013 - Future Emerging Technologies, Embodied Intelligence, under the grant agreements no. 231688 (Locomorph). We gratefully acknowledge the technical support of André Guignard, André Badertscher, Peter Brühlmeier, Philippe Voessler, and Manuel Leitos in the design and construction of the robot modules.

References

[1] N. A. Streitz, J. Geissler, and T. Holmer, "Roomware for cooperative buildings: Integrated design of architectural spaces and information spaces," in *CoBuild'98, Proceedings of*, p. 4, 1998.

[2] T. Fukuda, S. Nakagawa, Y. Kawauchi, and M. Buss, "Self organizing robots based on cell structures - cebot," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 145–150, 1988.

[3] H. Kurokawa, A. Kamimura, E. Yoshida, K. Tomita, S. Kokaji, and S. Murata, "M-TRAN II: metamorphosis from a four-legged walker to a caterpillar," in *IROS 2003. Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 3, pp. 2454 – 2459 vol.3, Oct. 2003.

[4] P. Pillai, J. Campbell, G. Kedia, S. Moudgal, and K. Sheth, ""a 3d fax machine based on claytronics"," in *ICRA 2006*, pp. 4728–4735, 2006.

[5] M. Asadpour, M. H. Z. Ashtiani, A. Sproewitz, and A. Ijspeert, "Graph signature for self-reconfiguration planning of modules with symmetry," in *IROS 2009*, (St. Louis), 2009.

[6] E. Yoshida, S. Murata, A. Kamimura, K. Tomita, H. Kurokawa, and S. Kokaji, "A motion planning method for a self-reconfigurable modular robot," in *Proceedings IROS 2001*, vol. 1, pp. 590 – 597, 2001.

[7] Z. Butler, K. Kotay, D. Rus, and K. Tomita, "Generic decentralized control for lattice-based self-reconfigurable robots," *IJRR*, vol. 23, pp. 919–937, sep 2004.

[8] P. White and M. Yim, "Scalable modular self-reconfigurable robots using external actuation," in *IROS 2007*, pp. 2773–2778, 2007.

[9] K. Stoy, "Using cellular automata and gradients to control self-reconfiguration," *Robotics and Autonomous Systems*, vol. 54, pp. 135–141, feb 2006.

[10] W. Shen, P. Will, A. Galstyan, and C. Chuong, "Hormone-inspired self-organization and distributed control of robotic swarms," *Autonomous Robots*, vol. 17, no. 1, pp. 93–105, 2004.

[11] J. V. Neumann, *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966.

[12] P. Varshavskaya, L. P. Kaelbling, and D. Rus, ""automated design of adaptive controllers for modular robots using reinforcement learning"," *IJRR*, vol. 27, pp. 505–526, mar 2008.

[13] R. Fitch and Z. Butler, "Million module march: Scalable locomotion for large self-reconfiguring robots," *The International Journal of Robotics Research*, vol. 27, pp. 331–343, mar 2008.

[14] K. D. Kotay and D. L. Rus, "Scalable parallel algorithm for configuration planning for self-reconfiguring robots," in *Sensor Fusion and Decentralized Control in Robotic Systems III* (G. T. McKee and P. S. Schenker, eds.), vol. 4196, (Boston, MA, USA), pp. 377–387, SPIE, oct 2000.

[15] D. Rus and M. Vona, "Crystalline robots: Self-reconfiguration with compressible unit modules," *Autonomous Robots*, vol. 10, no. 1, pp. 107–124, 2001.

[16] D. Dewey, M. Ashley-Rollman, M. D. Rosa, S. Goldstein, T. Mowry, S. Srinivasa, P. Pillai, and J. Campbell, "Generalizing metamodules to simplify planning in modular robotic systems," in *IROS 2008*, pp. 1338–1345, 2008.

[17] E. Mytilinaios, M. Desnoyer, D. Marcus, and H. Lipson, "Designed and evolved blueprints for physical self-replicating machines," *In Proc. of the 9 th Int. Conf. on the Simulation and Synthesis of Living Systems (Artificial Life IX). MIT*, vol. 2004, pp. 15–20, 2004.

[18] A. Sproewitz, A. Billard, P. Dillenbourg, and A. J. Ijspeert, ""roombots-mechanical design of self-reconfiguring modular robots for adaptive furniture"," in *ICRA 2009*, (Kobe, Japan), pp. 4259–4264, 2009.

[19] V. Zykov, E. Mytilinaios, B. Adams, and H. Lipson, "Self-reproducing machines," *Nature*, vol. 435, p. 163, 2005.

[20] Y. Terada and S. Murata, "Automatic modular assembly system and its distributed control," *IJRR*, vol. 27, pp. 445–462, mar 2008.

[21] A. Sproewitz, M. Asadpour, Y. Bourquin, and A. Ijspeert, "An active connection mechanism for modular self-reconfigurable robotic systems based on physical latching," in *ICRA 2008*, pp. 3508–3513, 2008.

[22] O. Michel, "Webots: Professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[23] P. Mudry, J. Ruffin, M. Ganguin, and G. Tempesti, "A hardware-software design framework for distributed cellular computing," in *Proceedings of the 8th international conference on Evolvable Systems: From Biology to Hardware*, p. 82, Springer, 2008.