EasySRRobot

An Easy-to-Build Self-Reconfigurable Robot with Optimized Design

Yu, Minjing; Liu, Yong-Jin; Wang, Charlie

DOI
10.1109/ROBIO.2017.8324563

Publication date
2017

Document Version
Accepted author manuscript

Published in
2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)

EasySRRobot: An Easy-to-Build Self-Reconfigurable Robot with Optimized Design

Minjing Yu, Yong-Jin Liu
National Lab for Information Science and Technology,
Department of Computer Science and Technology,
Tsinghua University, Beijing, China

Charlie C.L. Wang
Department of Design Engineering,
TU Delft Robotics Institute,
Delft University of Technology, The Netherlands

***Abstract*—*Self-reconfigurable modular robots* (SRRobot) that can change their shape and function in different environments according to different tasks have caught a lot of attention recently. Most existing prototypes use professional electronic components with relatively expensive cost and high barrier of fabrication. In this paper, we present a low-cost SSRobot with double-cube modules. Our system is easy-to-build even for novices as all electric components are off-the-shelf and the structural components in plastics are made by 3D printing. To have a better design of interior structures, we first construct a design space for all feasible solutions that satisfy the constraints of fabrication. Then, an optimized solution is found by an objective function incorporating the factors of space utilization, structural soundness and assembly complexity. Thirty EasySRRobot modules are manufactured and assembled. The functionality of our algorithm is demonstrated by comparing an optimized interior design with other two feasible designs and realizing different motions on an EasySRRobot with four modules.**

I. Introduction

A self-reconfigurable modular robot (SRRobot) is formed by a number of modules, which is physically independent and encapsulates a certain simple function. Complex tasks performed by a SRRobot are usually realized by the joint effort of all modules. A SRRobot can change their shape and therefore the function according to different tasks or environments – e.g., the EasySRRobot developed in our approach can transform into a snake shape for slithering into narrow tunnels / valleys or into a wheel shape for quickly traveling on flat terrains. Limited types of modules are used in a SRRobot so that self-repair is easier to be realized by replacing damaged modules. Due to above nice properties, the research of SRRobots has caught a lot of attention recently (ref. [1]).

Many types of SRRobots have been proposed in literature, among which the ones with double-cube modules have been widely used – e.g., M-TRAN series (ref. [2]–[4]), SuperBot [5], PolyBot [6], iMobot [7], Molecule [8], Shady [9] and Dtto [10]. Most of these double-cube modules are constructed by professional components including actuators, sensors, microprocessors, inter-module communication/power-transmission devices and electronic/magnetic connectors. As one of the most representative SSRobots, the M-TRAN III module [4] has the specifications as listed in Table I. The total cost of components used in M-TRAN III is $399.6, which is much more expensive than the components in our system (i.e., $31 in total). Moreover, M-TRAN III needs professional tools and skills for the fabrication and assembly. But our system can be easy even for novice to build. Noted that all existing design of double-cube modules including ours have similar exterior shapes and structures (i.e., each module consists of two semi-cylindrical boxes and a link). Similar functionality is provided to rotate each box independently around its axis by up to $\pm 90^\circ$.

Inspired by the success of Dtto [10], we present a low-cost and easy-to-build SRRobot in this paper, which has a price similar to Dtto but with both the hardware design and the interior-layout design optimized. Different from Dtto, all the on-board components used in our solution are off-the-shelf by following the design principle of using as-many-as-possible standardized components [11]. This hardware design simplifies the process of fabrication and meanwhile improves the reliability of our SSRobot. Even novice users can easily make the self-reconfigurable robot presented in this paper – this is why we name it as *EasySSRobot*. Another distinct feature of EasySRRobot is an optimized design of interior structures obtained by numerical computation, which are detailed below.

In this paper, we propose an algorithm to automatically find an optimal interior structure design for placing a given set of on-board components in a double-cube module. To achieve this goal, we characterize spatial relationships among on-board components by a set of design constraints and build a search space that contains all feasible interior structure designs. We propose a novel objective function that evaluates each feasible design by considering three evaluation criteria, including space utilization, structural soundness and assembly complexity. The optimal design is then obtained by minimizing the objective function in the search space, which is solved by the simulated annealing method. This is different from two existing parametric design techniques [12]:

- Propagation-based techniques, such as the OpenSCAD system [13], need users to set up a family of initial parameters and build a hierarchy of mathematical / geometric relations for generating a certain design;
- Constraint-based techniques, such as the SolidWorks system [14], usually requires users to set up and modify a set of non-linear constraints.

Neither of them can automatically obtain an optimal design.

To the best of our knowledge, the problem of automatic optimization of interior structure design in a double-cube module has not been studied before. To demonstrate the

(a) Functionality of a double-cube module

(b) Exterior structure of M-TRAN III [4]

Electronics

Connection mechanism

Mechanical parts

(c) Interior structure of M-TRAN III [4]

(d) Exterior and interior structures of Dtto v2 [10]

Fig. 1. All self-reconfigurable robots using double-cubic modules have similar exterior structures and similar functions that: each module consists of two semi-cylindrical boxes and a link, and each box can rotate independently around its axis by $\pm 90^\circ$ (see (a) for the design of our EasySRRobot). The male box (in green color) has slits on its three faces, and the female box (in yellow color) has hook cavities on its three faces. The double-cube modules used in M-TRAN III (see (b) and (c)) and Dtto v2 (see (d)) have different interior structures for placing on-board components, which lead to different placements of the link – the link of Dtto is placed closer to one side of the box than the other side.

effectiveness and usefulness of our algorithm, we compare the optimal design output from our algorithm with another two feasible designs and build a EasySRRobot prototype using the optimal design. Motions of two configurations (i.e., snake and wheel) are achieved, showing the working ability and effectiveness of the proposed EasySRRobot system.

II. Hardware and Module Design Principle

In this section, we first present the hardware components that are used in our EasySRRobot. After that, we summarize the design principles that are common in most double-cube modules (e.g., [2]–[6], [9], [10]).

TABLE I
Components used in M-TRAN III

Type	Description	Cost* ($)
Motor	Two HS-GM21-DSD/KS2 for the links and three HS-GM21-ALG for the connections (all from STL Japan);	203.0
Processors	One 32-bit HD64F7047 for main CPU, and two 16-bit HD64F3687 and one 16-bit HD64F3694 for other tasks (all from Rensus Corp);	60.8
Network	CAN bus with 1Mbps bandwidth;	0.1
Wireless	Bluetooth wireless modem (Zeevo ZV3001Z);	5.0
Sensor	10 IR proximity sensors, 13 IR diode (LNA2801A), 13 IR sensor (TAOS TSL260), and two acceleration sensor (ADXL202E);	90.7
Battery	Lithium-polymer (7.4V and 730mAh);	30.0
Structure	ABS for the links and polyacetal for other structural components.	10.0

*The costs are specified in US dollar ($).

TABLE II
Specification of on-board components in EasySRRobot

Item	QTY	Specification	Weight	Cost* ($)
HX1218D Servomotor	2	Max. torque 2.2 kg-cm	12g	13.2
SG90 Servomotor	3	Max. torque 1.8 kg-cm	9g	3.0
Control Circuit	1	Integrate with ATmega328P CPU HC-05 bluetooth module nRF24L01 transceiver IC	10g	32.3
Battery	1	Li-Po7.4v 500mAh	23.5g	5.6

*The costs are specified in US dollar ($).

A. Hardware

All the electric components of a double-cube module must be installed inside its two boxes, which are connected by a link (see Fig.1(a)). Following the strategy of M-TRAN series [2]–[4], we choose semi-cylindrical box as the module's shape in EasySRRobot. Each box can rotate around its own axis by $\pm 90^\circ$. For self-reconfigurable robots, hardware design must support efficient connection/disconnection operations between modules. In the literature of SRRobot, both magnetic (e.g., [2], [3]) and mechanical connections (e.g., [4], [10]) have been considered. It is reported in [4] that the connection/disconnection operation by a mechanical connector is more than fourteen times faster than a magnetic connector. Therefore, mechanical connections are used in our EasySRRobots. Specifically, two types of boxes are defined in each module (see also the illustration in Fig.1(a)):

- Male box: slits are made on its three planar faces so that hooks can be rotated out to latch the female box in other modules;
- Female box: hook cavities are produced on its three planar faces to be latched by the hook of a male box in the other module.

The mechanical connection between boxes designed in this way shows excellent stability in our experiments.

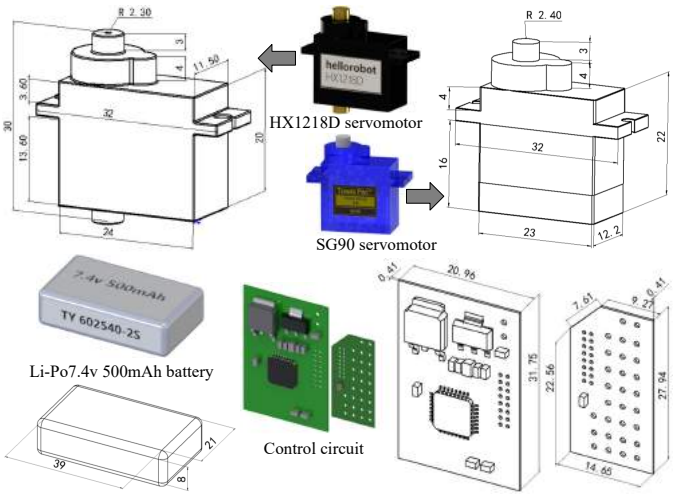


Fig. 2. The components used in EasySRRobot that have been listed in Table II. Specifically, the dimensions (Unit: mm) of their axis-aligned bounding boxes are: HX1218D servomotor ($32 \times 11.5 \times 30$), SG90 servomotor ($32 \times 12.2 \times 29$), the control circuits (one piece with $31.75 \times 20.96 \times 2.85$ and another piece with $27.94 \times 14.65 \times 1.71$) and the battery ($39 \times 21 \times 8$).

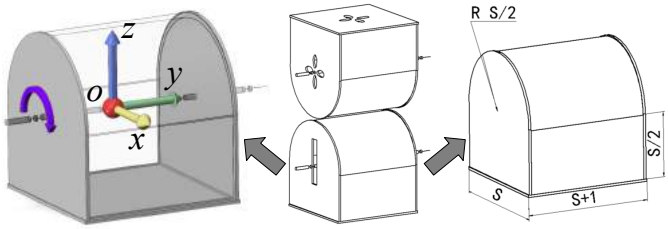
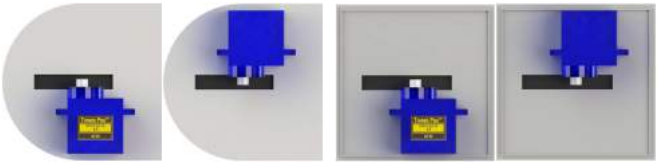


Fig. 3. The coordinate system (left) of the double-cube module in its rest pose (middle), where the male box and the female box are facing each other in the rest pose. Both boxes have the same shape and their dimensions are related to the parameters S and D.



(a) On a semi-disk face (b) On a rectangular face

Fig. 4. Constrained by the locations of slits, each SG90 servomotor c_{SG_i} can only have two candidate locations $\{\Theta_{SG_i,1}, \Theta_{SG_i,2}\}$ (above or below the slit) on one of the three faces with slits in the male box.

Comparing to the existing double-cubic SRRobot (e.g., M-TRAN and Dtto), the modular design concept has been employed in our hardware design. Only severn components (with four different types) are used, where the major components are five actuators including two HX1218D servomotors for rotating the male and the female boxes respectively and three SG90 servomotors for driving the hooks in the male box. These actuators are driven by a control circuit built on the Arduino MCU with ATmega328P CPU. The control circuit also integrates a bluetooth module (HC-05) and a transceiver IC (nRF24L01) for communication. As a result, every double-cube module can communicate with a host PC through the bluetooth module and two modules can communicate with each other via the transceiver IC. All these components are compatible with Arduino - an open-source electronics prototyping platform[1]. The power for each dual-cubic module is supplied by a Li-Po7.4v 500mAh battery. Table II lists the on-board components used in the hardware design of our EasySRRobot, and their dimensions are shown in Fig.2. Note that, all these components and their drivers are off-the-shelf, which greatly reduces the cost and simplifies the steps of fabrication.

B. Spatial Relationships and Design Principle

We now start to analyze design constraints that reflect spatial relationships between components to be placed inside a double-cube module. To specify these constraints, we first construct a coordinate system in the double-cube module (see also Fig.3 for an illustration):

- the origin o is set to be located at the center of the male box;
- the rotation axes of both the male and the female boxes are set along the y-direction;
- the cylindrical surface of the male box is facing up - i.e., along the $+z$ direction;
- in the rest pose of the module, the cylindrical surface of the female box is facing down;
- the centre of female box is located at $(0, 0, S)$ with S being the distance between two boxes rotation axes.

Note that, all the design constraints discussed below are proposed according to the configuration of rest pose.

Both the male and female boxes in a module have the same shape with dimensions S, $S+1$ and $S/2$ along the x, y and z directions respectively (see the right of Fig.3). To reduce the space of search in design optimization, quantization is applied to the value of S to make it as one of the four values sampled between the feasible maximum and minimum. And a fixed value d is chosen according to experiments. More details can be found in Section III as the step of paramterization. Now we start to discuss the design constraints.

For each on-board component c_i, we denote its axis-aligned bounding box as $B(c_i)$. Then, the pose of c_i can be specified by $\Theta(c_i) = (x_i, y_i, z_i, \alpha_i, \beta_i)$, where (x_i, y_i, z_i) is the position of $B(c_i)$s centre and (α_i, β_i) gives two spherical angles indicating its orientation. Let Ω_M and Ω_F be the spaces enclosed by the boundary surfaces of male and female boxes. The first design constraint is about enclosure.

Constraint 1: The axis-aligned bounding box $B(c_i)$ for each on-board component c_i in a designed pose $\Theta(c_i)$ must be inside $\Omega_M \cup \Omega_F$.

In our hardware design, three SG90 servomotors $\{c_{SG_1}, c_{SG_2}, c_{SG_3}\}$ are used to drive the hooks in the male box. They need to attached onto faces with slits –

[1]http://www.arduino.cc/

Fig. 5. Examples of possible orientations of two HX1218D servomotors for driving the link. Rotation axes of servomotors coincide with the rotation axes of boxes, and the link connecting two servomotors must be perpendicular to rotation axes. In our design, the link is hollowed such that electric wires connecting the male and female boxes can be placed inside.

one for each slit, which is located in the middle of the corresponding face (see Fig.4). The second design constraint is for locating the actuators for connection/disconnection.

Constraint 2: A SG90 servomotor, c_{SG_i}, for driving a hook for connection can only be placed at one of the two locations $\Gamma_{SG_i} = \{\Theta_{SG_i,1}, \Theta_{SG_i,2}\}$, $i = 1, 2, 3$.

Two HX1218D servomotors $\{c_{HX_1}, c_{HX_2}\}$ are used to drive the link connecting two boxes in a module, where each box contains a HX1218D servomotor with its rotation axis coinciding with the rotation axis of the box (i.e., the y-axis in rest pose). Moreover, the line connecting the centers o_{HX_1} and o_{HX_2} of two servomotors should be perpendicular to y-axis to enable the rotation of a link to both boxes. These design factors are summarized into the following two constraints.

Constraint 3: All the possible poses of a HX1218D servomotor c_{HX_j} in a male / female box should be mapped from $\mathbb{R}^5$ into $\mathbb{R}^2 = (y_{HX_j}, \alpha_{HX_j})$ because x_{HX_j}, z_{HX_j} and β_{HX_j} are fixed as constants to enable the rotation between boxes around y-axis.

Constraint 4: To achieve a stable cooperation in rotation, the constraint with the same value of y_{HX_j} is also imposed to the two HX1218D servomotors c_{HX_1} and c_{HX_2}.

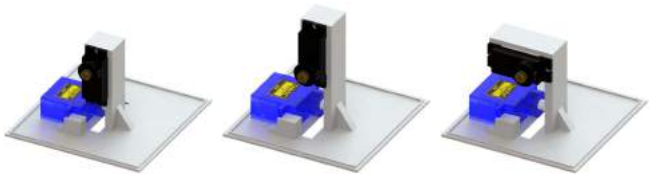
Each HX1218D servomotor has two ends along its rotation axis that can be mounted to the link, we design a weight balanced link during rotation as shown in Fig.5. Besides of above actuators, the placement of two control circuits $\{c_{ctrl_1}, c_{ctrl_2}\}$ and the battery $c_{battery}$ will only follow *Constraint 1* – located in Γ_{ctrl_1}, Γ_{ctrl_2} and $\Gamma_{battery}$ respectively, which are subsets of $\mathbb{R}^5 = (x_i, y_i, z_i, \alpha_i, \beta_i)$, $i = ctrl_1, ctrl_2, battery$.

III. Interior Structure Optimization

The design of interior structure is optimized by changing the poses of components inside a double-cube module according to the constraints introduced in Section II-B.

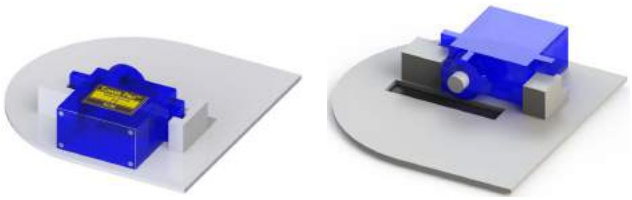
First of all, we determine the minimal and the maximal dimensions of the boxes with reference to the dimensions of components. As a result, $S \in [68mm, 81mm]$ is obtained. In order to reduce the searching time of optimization, this feasible range of S's value is quantized into four choices in our algorithm that is $S = 68mm, 72mm, 76mm$ or $81mm$. Second, for each on-board component c_i, our algorithm will automatically add a structural socket $S(c_i)$ in $\Omega_M(s) \cup \Omega_F(s)$ to fix it. Examples of such sockets can be found in Fig.6 and 7 for the servomotors. The socket for battery is similar to Fig.6, and the sockets for control circuits are similar to the examples shown in Fig.7. According to the constaints, each on-board component c_i has its own feasible space of poses Γ_i. Therefore, the design space can be defined by the Cartesian product of all these feasible spaces as $\Gamma = \prod_i \Gamma_i$. For any point $\mathbf{p} \in \Gamma$, our algorithm will check whether it is collision-free together with the automatically added sockets.

(a) Three examples of sockets in the female box



(b) Three examples of sockets in the male box

Fig. 6. Some examples of using additional sockets to fix the HX1218D servomotors according to the poses shown in Fig.5.



(a) On the semi-disk face

(b) On the rectangular face

Fig. 7. Sockets of SG90 servomotors for the two candidate locations shown in Fig.4.

A. Objective Function

The objective function consists of three factors to be considered in the design, including structural soundness, space utilization and assembly complexity, each of which results in a term in $F(\mathbf{p})$.

$$F(\mathbf{p}) = w_{str}F_{str}(\mathbf{p}) + w_{spa}F_{spa}(\mathbf{p}) + w_{asm}F_{asm}(\mathbf{p}) \quad (1)$$

where w_{str}, w_{spa} and w_{asm} are non-negative weights. According to our experimental tests, $w_{str} = 10.0$, $w_{spa} = 1.0$ and $w_{asm} = 1.5$ are used to balance the trade-off between different terms.

Structural Soundness Any collision-free configuration $\mathbf{p} \in \Gamma$ is corresponding to a feasible design of interior structures including the shape of box containers, the sockets and the on-board components. We then take a *Finite Element Analysis* (FEA) on this feasible interior structure by using the publicly available library[2]: *SfePy*. Gravity and torques from servomotors are set as external loadings in this analysis. With $\sigma_{\max}$ denoting the maximal stress, the score of this structure can be evaluated by

$$F_{str}(\mathbf{p}) = \frac{\sigma_{\max}}{\sigma_{ref}} \tag{2}$$

where σ_{ref} is a reference stress. The reference stress is computed as an average of maximal stresses from five manually designed feasible interior structures. The smaller $F_{str}(\mathbf{p})$, the better structural soundness a design is.

Space Utilization A good design needs to be compact – i.e., fully using the space provided by the boxes in a module. The efficiency of space utilization is measured by the ratio of volume used by on-board components.

$$F_{spa}(\mathbf{p}) = \frac{V(\Omega_M) + V(\Omega_F)}{\sum_{i=1}^{8} V(c_i)}, \tag{3}$$

where $V(c_i)$ is the volume of component c_i, $V(\Omega_M)$ and $V(\Omega_F)$ are volumes of male and female boxes respectively. In the literature of robotics (e.g., [4]), the power-to-weight ratio is an important metric for actuators. Given that the power of servomotors is fixed, the power-to-weight is inversely proportional to $F_{spa}(\cdot)$ – i.e., the smaller $F_{spa}(\cdot)$, the better space utilization is achieved by a design.

Assembly Complexity A collision-free placement of all on-board components does not mean that they can be validly assembled. Even if it is able to assemble, different assembly sequences produce different assembly complexities. A good structural design should give a low assembly complexity, i.e., easy to assemble and disassemble. Given a collision-free placement $\mathbf{p} \in \Gamma$, we use the generic algorithm [15] to compute an optimal assembly sequence Υ for all components $(c_1, c_2, \cdots, c_n)$. Also, the complexity of a valid assembly sequence is usually measured by the number of re-orientations needed during the assembly. Details about how to evaluate the validity of an assembly sequence and its complex can be found in Appendix. Here, the objective function of assembly complexity for Υ is defined as

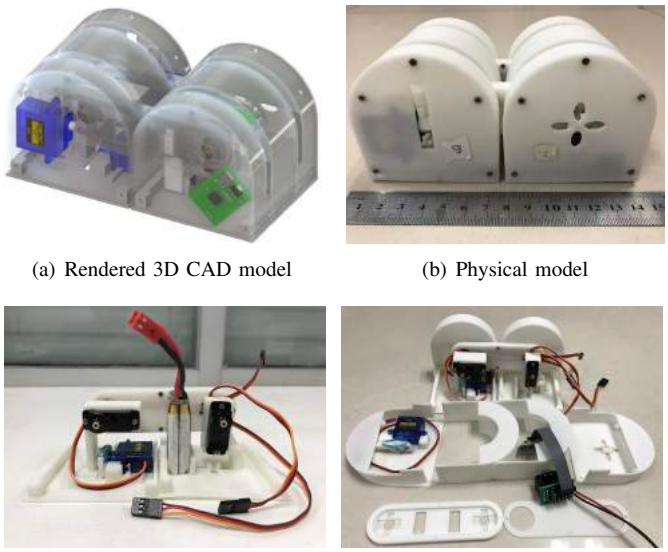
$$F_{asm}(\mathbf{p}) = \begin{cases} 10^6 & \text{when } \Upsilon \text{ is invalid} \\ n_{ort}(\Upsilon) & \text{otherwise} \end{cases} \tag{4}$$

where n_{ort} is the number of re-orientations in a valid assembly sequence Υ.

B. Simulated Annealing

Finding an optimal solution to minimize the objective function (1) in the search space $\Gamma = \bigcap_i \Gamma_i$ is a mixed combinatorial and continuous optimization problem because that the subspaces Γ_i ($i = SG_1, SG_2, SG_3$) are discrete and other subspaces are continuous. We adopt the *Simulated Annealing* (SA) method [16], [17] to solve it.

[2]http://sfepy.org/doc-devel/index.html



(a) Rendered 3D CAD model (b) Physical model

(c) Two intermediate steps during the fabrication process

Fig. 8. The optimized design generated by our approach and its physical realization.

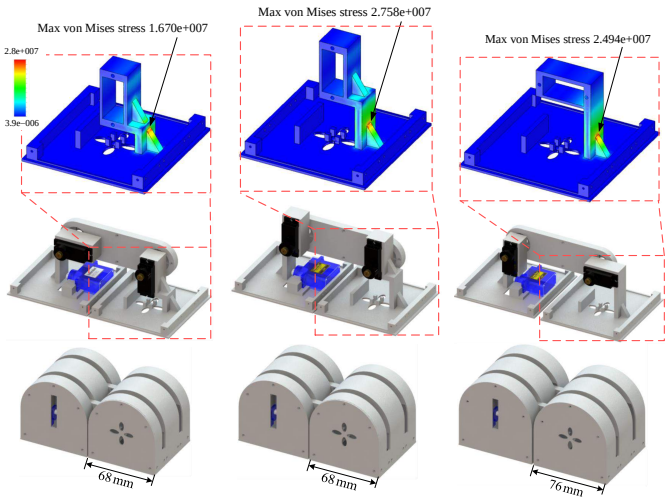
Simulated annealing is probabilistic technique that can efficiently approximate the global optimum. To apply the SA technique, we first generate a random solution $\mathbf{q}$ in Γ and evaluate its cost $F(\mathbf{q})$ using the objective function (Eq.(1)). A point $\mathbf{p} \in \Gamma$ is called a feasible solution $\mathbf{q}$ if $\mathbf{p}$ corresponds to a collision-free placement of all the on-board components. Then we generate a random neighboring solution $\mathbf{q}'$ and compute the cost $F(\mathbf{q}')$. After that, the Metropolis acceptance criterion is adopted to determine whether the system status can move from the current solution $\mathbf{q}$ to the candidate solution $\mathbf{q}'$ with the acceptance probability $P(\mathbf{q}, \mathbf{q}')$:

$$P(\mathbf{q}, \mathbf{q}') = \begin{cases} \exp(-\frac{F(\mathbf{q}') - F(\mathbf{q})}{T}) & \text{if } F(\mathbf{q}') > F(\mathbf{q}) \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

Specifically, if the value of $P(\mathbf{q}, \mathbf{q}')$ is greater than a randomly selected threshold in $[0, 1]$, we will move from $\mathbf{q}$ to $\mathbf{q}'$. Here, T is the temperature parameter to control the speed of SA computation. Starting from $T = 100$, the temperature T is reduced by 5% after each iteration. During the Monte Carlo procedure of system status movement, we always keep a best solution – the configuration $\mathbf{q}$ that gives the minimal value of $F(\mathbf{q})$. It was shown [16] that if the temperature was cooled slowly, a global minimum can be found. The system iteratively updates the solution until a specified iteration number is reached or the cost does not decrease anymore.

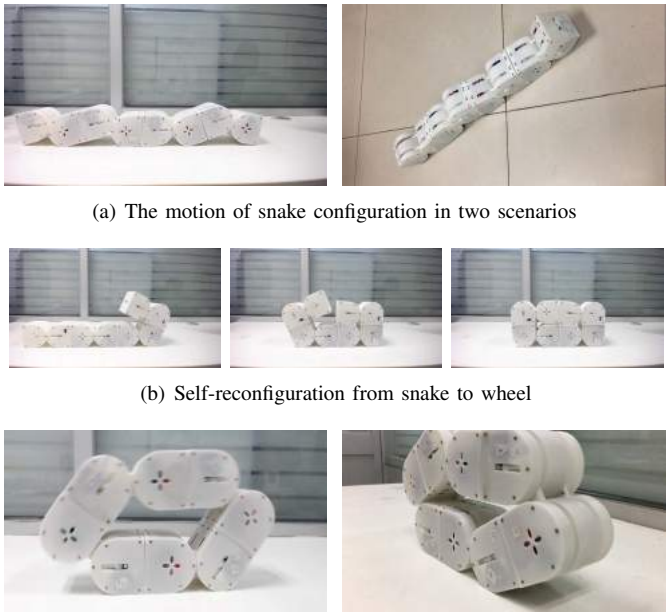
IV. Experimental Results

The proposed design method has been implemented on a physical self-reconfigurable robot – our EasySSRobot as shown in Fig.8 and 9. We implement the proposed design



(a) Optimal design (b) Feasible design I (c) Feasible design II

Fig. 9. The comparison of the optimized design from our approach (a) with two other manually designed feasible solution (b) and (c). The top row shows the stress distribution on the supporting structure of the HX1218D servomotor in the female box, in which the maximum stress appears. The middle row shows interior details – to better illustrate the positions of HX1218D servomotors (black), the front link component is hidden here. The bottom row shows the overall appearance of the three designs.



(a) The motion of snake configuration in two scenarios

(b) Self-reconfiguration from snake to wheel

(c) The motion of wheel configuration in two scenarios

Fig. 10. The screenshots of snake and wheel configurations, and a self-reconfiguration – see also the accompanying demo video for more details.

optimization algorithm in C++ and test it on a PC with an Intel E5-2650 CPU@2.60GHz and 64GB RAM. Since structural soundness is more important than space utilization and assembly complexity, we use $w_{str} = 10.0$, $w_{spa} = 1.0$ and $w_{asm} = 1.0$ as weights in the objective function (Eq.(1)) for all examples shown in this paper. The algorithm takes 24 minutes to compute an optimized design as shown in Fig.8(a). Thirty modules were manufactured for EasySRRobot with the aid of 3D printing (see Fig.8(b) for an outlook of the physical model). Since all on-board components are off-the-shelf and the complexity of assembly has been considered in the algorithm, assembling these components into a module of EasySRRobot is easy – as shown in Fig.8(c).

We have also compared the optimized design generated by our algorithm with two manually designed feasible solutions in Fig.9. In the optimized design (Fig.9(a)), two HX1218D servomotors (displayed in black color) are perpendicular to each other. Differently, in feasible design I (Figure 9(b)), these motors (black) are parallel to each other. Although having the same box size, the maximum stress ($\sigma_{\max} = 2.758 \times 10^7$) of the feasible design I is much larger than the optimized design with $\sigma_{\max} = 1.670 \times 10^7$ in the unit of N/m^2. In the feasible design II (Fig.9(c)), the orientations of motors (black) are also perpendicular but in an inverse way of ours. Due to the potential collision between the HX1218D servomotor (black) and a SG90 servomotor (blue) in the male box, both the box size ($S = 76mm$) and the maximum stress ($\sigma_{\max} = 2.494 \times 10^7$) of the feasible design II are larger than our optimized design ($S = 68mm$ and $\sigma_{\max} = 1.670 \times 10^7$).

In our physical experimental tests, the functionality of self-reconfiguration has been examined on a fabrication according to our optimized design – EasySRRobot. Figure 10 shows the screenshots of two configurations (i.e., snake and wheel modes) and a self-reconfiguration transforming from a snake to a wheel robot, which is autonomously functioned by EasySRRobot without human intervention. The motion details of these two and more configurations are presented in the accompanying demo video: http://47.89.51.189/liuyj/EasySRRobot-demo.zip.

V. Conclusion

In this paper, we present a low-cost and easy-to-build self-reconfigurable modular robot called EasySRRobot. To optimized the design of EasySRRobot by using a set of off-the-shelf components, we characterize the design principle of double-cube modules in quantitative constraints and propose an objective function to evaluate any feasible placement of on-board components. All the feasible placements are characterized in a design space Γ. Our objective function considers three criteria of an optimal design, including structural soundness, space utilization and assembly complexity. The simulated annealing technique is applied to minimize the objective function in the design space Γ. The interior structure generated by our algorithm has been implemented in an EasySRRobot as carrier and two configurations (snake and wheel) of EasySRRobot are presented to demonstrate the effectiveness of our robot and the proposed algorithm for design optimization.

Acknowledgment

This work was supported by the National Key Research and Development Plan (2016YFB1001202), Royal Society-Newton Advanced Fellowship and the Natural Science Foun-

dation of China (61521002, 61432003, 61661130156). C.C.L. Wang is also partially supported by the open project fund of Central South University, Hunan, China.

REFERENCES

[1] K. Stoy, D. Brandt, and D. J. Christensen, *Self-Reconfigurable Robots: An Introduction*. The MIT Press, 2010.

[2] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-tran: self-reconfigurable modular robotic system," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 4, pp. 431–441, 2002.

[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 *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003*, 2003, pp. 2454–2459.

[4] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata, "Distributed self-reconfiguration of M-TRAN III modular robotic system," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 373–386, 2008.

[5] B. Salemi, M. Moll, and W. Shen, "SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2006*, 2006, pp. 3636–3641.

[6] M. Yim, D. Duff, and K. Roufas, "Polybot: A modular reconfigurable robot," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000*, 2000, pp. 514–520.

[7] iMobot, *http://spectrum.ieee.org/automaton/robotics/diy/imobot-brings-robot-modules-to-modular-robots*, 2011.

[8] K. Kotay, D. Rus, M. Vona, and C. D. McGray, "The self-reconfiguring robotic molecule," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '98)*, 1998, pp. 424–431.

[9] Shady, *https://groups.csail.mit.edu/drl/Shady/shady.htm*, 2016.

[10] Dtto_Modular_Robot_V2.0, *https://hackaday.io/project/9976-dtto-v20-modular-robot*, 2016.

[11] T.-H. Kwok and C. C. Wang, "Shape optimization for human-centric products with standardized components," *Computer-Aided Design*, vol. 52, pp. 40–50, 2014.

[12] R. Woodbury, *Elements of Parametric Design*. Routledge, 2010.

[13] OpenSCAD, *https://www.openscad.org*, 2017.

[14] SOLIDWORKS, *https://www.solidworks.com*, 2017.

[15] S.-F. Chen and Y.-J. Liu, "An adaptive genetic assembly-sequence planner," *International Journal of Computer Integrated Manufacturing*, vol. 14, no. 5, pp. 489–500, 2001.

[16] P. J. M. Laarhoven and E. H. L. Aarts, Eds., *Simulated Annealing: Theory and Applications*. Norwell, MA, USA: Kluwer Academic Publishers, 1987.

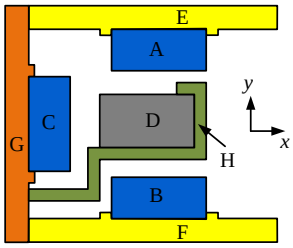
[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing, 3rd Edition*. Cambridge University Press, 2007.

APPENDIX: EVALUATION OF ASSEMBLY SEQUENCE

To automatically check the validity of an assembly sequence and compute the complexity of an assembly sequence, we use an *assembly matrix* representation $\mathbf{M}$ to describe the geometric constraints between components in an assembly (ref. [15]). $\mathbf{M}$ for n components is a $n \times n$ matrix, where its (i, j) entry stores directions along which the component c_i can be assembled without colliding with the component c_j. Note that, the components are allowed to translate along x-, y- and z-axes. An example assembly in 2D and its corresponding $\mathbf{M}$ can be found in Fig.11. For instance, at the entry $(1, 2) = (A, B)$, the value $(\pm x - y)$ means that A can be assembled along $(\pm x)$ or $(-y)$ directions without colliding to B.

Validity: Given an assembly sequence $\Upsilon = (c_1, c_2, \cdots, c_n)$,

$$\mathbf{V}(c_i) = \bigcap_{j<i} \mathbf{M}(i, j) \tag{6}$$



Assembly matrix

	A	B	C	D	E	F	G	H
A	\	±x−y	±x±y	±x−y	+y	±x−y	−x±y	±x−y
B	±x+y	\	±x±y	±x+y	±x+y	−y	−x±y	−x+y
C	±x±y	±x±y	\	+x±y	±x+y	±x−y	−x	+x−y
D	±x+y	±x−y	−x±y	\	±x+y	±x−y	−x±y	+x
E	−y	±x−y	±x−y	±x−y	\	±x−y	−x−y	±x−y
F	±x+y	+y	±x+y	±x+y	±x+y	\	−x+y	±x+y
G	+x±y	+x±y	+x	+x±y	+x+y	+x−y	\	+x−y
H	±x+y	+x−y	−x+y	−x	±x+y	±x−y	−x+y	\

Fig. 11. A 2D structure to be assembled (left) and its assembly matrix $\mathbf{M}$ (right). The validity and the complexity of an assembly sequence can be evaluated with the help of $\mathbf{M}$.

can be used to compute the set of possible assembly directions of each c_i according to this sequence, where $\bigcap$ is the operator of set intersection. If there is any $\mathbf{V}(c_i) = \emptyset$, the assembly sequence Υ is invalid. For example the case shown in Fig.11, when $\Upsilon = (G, C, H, D, A, B, E, F)$ we have

$$\begin{aligned}\mathbf{V}(D) &= \mathbf{M}(D, G) \bigcap \mathbf{M}(D, C) \bigcap \mathbf{M}(D, H) \\ &= (-x, \pm y) \bigcap (-x, \pm y) \bigcap (+x) = \emptyset.\end{aligned}$$

Then, Υ is not a valid assembly sequence.

Complexity: Given $\Upsilon = (c_1, c_2, \cdots, c_n)$ as a valid assembly sequence, it means that we have $\mathbf{V}(c_i) \neq \emptyset$ for every $c_i \in \Upsilon$. Then, the complexity of an assembly sequence can be defined with the help of number of re-orientation operations. Specifically,

- if $\bigcap_{k=i}^{j} \mathbf{V}(k) \neq \emptyset$, no re-orientation is needed during the assembly of $c_i, \cdots, c_j$;
- if $\bigcap_{k=i}^{j} \mathbf{V}(k) \neq \emptyset$ and $\bigcap_{k=i}^{j+1} \mathbf{V}(k) = \emptyset$, one re-orientation applied to the already assembled components is needed for assembling c_{j+1}.

Note that, after the reorientation and then assembling c_{j+1}, evaluation for the need of orientation restarts from c_{j+2} – i.e., whether $\bigcap_{k=j+2}^{m} \mathbf{V}(k)$ is empty will be computed. For example, $\Upsilon = (G, C, A, E, D, H, B, F)$ is a valid assembly sequence for the 2D case shown in Fig.11 that needs two re-orientations.

- In this case,
$$\mathbf{V}(G) \bigcap \mathbf{V}(C) \bigcap \mathbf{V}(A) = (-x)$$
and
$$\mathbf{V}(G) \bigcap \mathbf{V}(C) \bigcap \mathbf{V}(A) \bigcap \mathbf{V}(E) = \emptyset,$$
the first re-orientation is needed for assembling E.
- After that,
$$\mathbf{V}(E) = (-y) \text{ and } \mathbf{V}(E) \bigcap \mathbf{V}(D) = \emptyset,$$
the second re-orientation is needed for assembling D.

This computation is based on the values of $\mathbf{V}(\cdot)$ shown below.

Assembly Sequence: $\Upsilon = (G, C, A, E, D, H, B, F)$		
$\mathbf{V}(G) = (\pm x, \pm y)$	$\mathbf{V}(C) = (-x)$	$\mathbf{V}(A) = (-x, \pm y)$
$\mathbf{V}(E) = (-y)$	$\mathbf{V}(D) = (-x, +y)$	$\mathbf{V}(H) = (-x)$
$\mathbf{V}(B) = (-x, +y)$	$\mathbf{V}(F) = (+y)$	