Playdough to Roombots: Towards a Novel Tangible User Interface for Self-reconfigurable Modular Robots*

Mehmet Mutlu[1,2], Simon Hauser[1], Alexandre Bernardino[2] and Auke Ijspeert[1]

***Abstract*— One of the main strengths of self-reconfigurable modular robots (SRMR) is their ability to shape-shift and dynamically change their morphology. In the case of our SRMR system "Roombots", these shapes can be quite arbitrary for a great variety of tasks while the major utility is envisioned to be self-reconfigurable furniture. As such, the ideas and inspirations from users quickly need to be translated into the final Roombots shape. This involves a multitude of separate processes and - most importantly - requires an intuitive user interface. Our current approach led to the development of a tangible user interface (TUI) which involves 3D-scanning of a shape formed by modeling clay and the necessary steps to prepare the digitized model to be formed by Roombots. The system is able to generate a solution in less than two minutes for our target use as demonstrated with various examples.**

***Index Terms*— tangible user interface, self-reconfigurable modular robots, deformable material, shape formation**

I. INTRODUCTION

Roomboots is an SRMR system designed in Biorobotics Laboratory, EPFL [1]. The mechanical design, control of self-reconfiguration and user interfaces are three main work divisions of Roombots. One of the grand challenges [2] for the intended usage of structures built with SRMR is the method of how the inspiration for a structure by a user is translated into an real-world representation with SRMR. This process is illustrated in Fig. 1. We formally split the full process into five subprocesses: (i) expression, where an idea of a user takes shape in the real world, (ii) digitization, where this idea gets digitized and put into a PC, (iii) abstraction, where the raw digital representation is post-processed into a representation that can be used by an SRMR system, (iv) planning where algorithms produce an instruction plan that creates the desired shape in the form of a building sequence and finally (v) formation, where the SRMR form into the initial inspiration in the real world by following the building plan provided by the planning step. It is important to notice that only the first and the last processes take place in the real world and only they are usually able to provide any other than visual feedback (e.g. haptic feedback) which is the primary type of feedback in the digital representation (Fig. 1).

*This work was supported by the Fundação para a Ciência e Tecnologia (FCT) agency of Portugal under the contract number PD/BD/105781/2014 and project grant AHA-CMUPERI/HCI/0046/2013 and the Swiss National Science Foundation (Project 153299).

[1]M. Mutlu, S. Hauser and A. Ijspeert are with the Biorobotics Laboratory, School of Engineering and Institute of Bioengineering, École Polytechnique Fédérale de Lausanne, Switzerland mehmet.mutlu at epfl.ch

[2]M. Mutlu and A. Bernardino are with the Electrical and Computer Engineering of the Faculty of Engineering at IST, Instituto Superior Técnico, Lisbon, Portugal

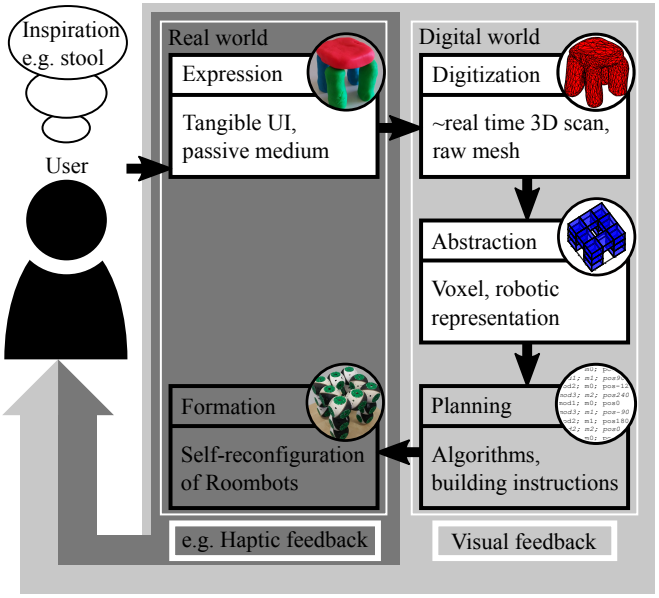


Fig. 1. The cycle from an inspiration to the final Roombots shape. The inspiration first is expressed in the real world. A digitization step transforms it into the digital world where the processing steps abstraction and planning can take place. At last, the inspiration takes shape with the SRMR by the formation.

Earlier works on Roombots focus mostly on only single processes of Fig. 1. In [3], a user utilizes a tablet PC and augmented reality to virtually place pre-defined structures into a living room; the inspiration is never expressed in real world and digitization does not take place since the abstraction step is directly performed manually. [4] proposes a direct human robot interaction (HRI) to control the location of modules in discrete grid environment in the real world, whereas [5] extends this idea to the continuous space; the expression and digitization steps are skipped in both cases. In contrast, much stand-alone work has been done in the planning step (e.g. [6]), where a selected set of abstracted structures have been provided to reconfiguration algorithms. The formation step has not yet been the primary focus and is subject of ongoing hardware development but ultimately necessary to demonstrate the complete cycle.

This work focuses on the three aspects expression, digitization and abstraction as a whole. In particular, we seek a method that is able to rapidly produce the abstracted representation of an idea; as one of the main strengths of SRMR is their ability to shape-shift and create (almost) arbitrary structures, a fast abstraction of the inspiration is necessary to allow a purposeful interaction with an SRMR-

system. We present a method that combines the relatively novel technology of 3D-scanning with an existing algorithmic approach (DFS) with adjustments to autonomously generate the building instructions for Roombots, bringing us one step closer to the vision of Roombots to be used as "fast-protyping" user-created furniture. At last, the formation (self-reconfiguration) step is subject of ongoing hardware development and will not be discussed in this work.

II. ANALYSIS OF THE PROBLEM

The main purpose of this study is to create a user friendly interface to generate a final *shape of structures* to be self assembled by Roombots. The interface should come up with a construction plan of the a desired arbitrary structure. For this, each of the three processes (see Fig. 1) leading to the processing step are explained in detail below, motivating our choice of each of the used methods. The explanation of each method can be found in further sections.

A. Expression: Modeling clay

An inspiration can be expressed in the real world in many different ways. It can be in the form of a drawing, a spoken or written description or even as a pantomimic gesture. As the purpose of Roombots is to form 3-dimensional shapes, expressing the inspiration also in a 3-dimensional way seemed to be the most intuitive approach. However, the size of the structures formed by Roombots is roughly in the order of magnitude of a human. Hence, a method to form a *model* of the inspiration was desired: a Tangible User Interface (TUI) [7] [8]. There exist diverse interaction media proposed in the literature that could benefit a TUI. For instance, computational building blocks in [9] and active cubes proposed in [10] can immediately transform the physical structure that a user is building into the digital domain as a 3D voxel array. Furthermore, [11] offers a construction method with kinetic memory which includes the ability of a user to input the shape of the structure as well as the expected motion. The Roombots is a complex robotic system due to the nature of SRMRs. In order to simplify the interaction, we inclined for a passive medium. In this work, the passive medium can be anything (LEGO® blocks, wood pieces etc.) that allows creation of structures that can be replicated with Roombots. Modeling clays are widely used for 3D shape modeling in games, e.g. Cranium® and Cluzzle®, or professional activities like architecture or art. Play-Doh® or similar modeling clays are very easy to shape and made for kids to create 3D shapes. Modeling clay is also one of the most studied tangible user interface (TUI) material used in the literature. For instance, [12], [13] use the modeling clay for fascinating landscape analysis and digital 3D design. [14] creates electronic circuits using conductive and insulating playdough. Another usage is explained in [15] where modeling clay is used to design an ergonomic PC mouse.

B. Digitization: 3D scanning

Having a passive interaction medium requires an additional step to digitize the shape information that is formed by the playdough (note that if the inspiration is directly digitized e.g. by modeling a structure in a PC, the expression step can be skipped). In this regard, we rely on commercial scanning solutions, which are becoming widely available and cheaper by the day, to convert the real object to the digital data. The result of the 3D-scanning process is a mesh grid that approximates the shape of an object with surface triangles.

C. Abstraction: Voxels and Roombots

For representing the 3D information, we use a voxel representation. Voxels are commonly used to represent SRMRs, particularly lattice types like Roombots. Example usage of cubic SRMR representation can be seen in [16], [17] and [18]. The quality of voxelization greatly depends on the voxel resolution and generally improves with a higher resolution, for which small and many voxels are required. However, Roombots modules are relatively big such that a representation in which a voxel corresponds to a half Roombots module would result in big voxels and low resolution. Low resolution voxelization is likely to result in undesired information loss and a voxel array that possibly is not resembling the original shape anymore. Nevertheless, it is possible to preserve the rough shape information even with big/coarse voxels when voxelization is done in a smart way. An advertisement of LEGO® is cleverly illustrating this challenge in Fig. 2.

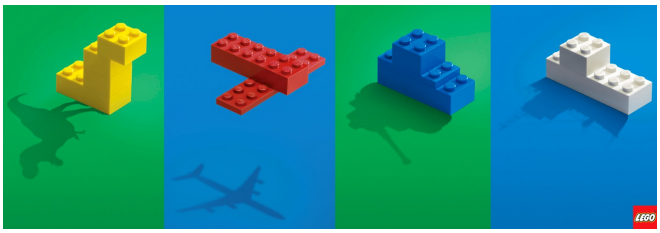


Fig. 2. The inspirational LEGO® advertisement emphasizing the shape representation capabilities of primitive construction bricks.

D. Constraints and simplifications

Some shapes cannot be built with Roombots due to their structure which consists of two consecutive cubic/spheric shape. For instance, the **T** shaped block (in its elementary, symmetric form) in the famous Tetris® game is physically impossible to build with Roombots modules.

Each Roombots module has three degrees of freedom (DOF) that can continuously rotate and two active connection mechanisms (ACM) that allows modules to connect to each other or to an environment equipped with connection plates [1]. Such motion capabilities allow a single module to locomote by itself and gives it functional flexibility. However, the same blessing of motion capabilities turn into a curse of dimensionality in a search problem. Nevertheless, most of the furniture-like structures made out of Roombots that have been demonstrated so far do not make use of DOFs, i.e. each joint of a Roombots module is set to zero degrees and the module resembles two consecutive voxels. Even though the continuous rotation of each DOF increases the number of possible structures that can be built with Roombots, in

this work we use modules as if they are construction bricks. Hence, construction is done only in orthogonal axes.

III. SOLUTION METHOD

The construction of structures with a shape given by the playdough is a multi stage process that is almost completely autonomous. Once the user shapes the playdough, the following stages are (i) 3D scanning, (ii) mesh pre-processing, (iii) voxelization, (iv) initial module placement (v) construction search and (vi) the user feedback.

A. 3D Scanning of playdough

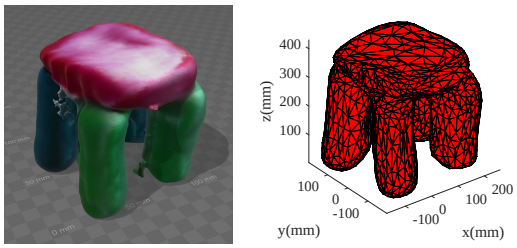
3D scanning is needed to capture the information sculpted into playdough. There is a deep research literature on 3D scanning focusing on many different methodologies. An early study in [19] captures 3D mesh in real time using a camera. In a more recent study, [20] shows how to fuse color and depth information to have high quality 3D scans. [21] suggests a method to scan modeling clay in 2.5D in real time. There are also commercial 4D scanners (3D scanners that can capture temporal changes). However, they are usually expensive systems. [22] offers an open source and one of the most economical instantaneous 3D scanners.

It is possible to replicate the system in [22] in a smaller scale for smaller objects. However, developing a 3D scanning system is out of scope of this work. We picked the cheapest commercially available solution: a 3D scanner from XYZprinting® for 200$. It is using an Intel® RealSense™F200 camera to obtain depth and color images. The Intel RealSense SDK and 3D Scan tool (11.0.27.8892) is used for 3D scanning. As a result, we compromised on real-time scanning for the sake of cost efficiency. Nevertheless, a playdough structure similar to the one shown in Fig. 1 can be scanned in approximately 45 to 90 seconds.

B. Pre-Processing of Mesh

3D scanned meshes need a quick and almost automatized preprocessing. The initial meshes are solidified (to obtain closed surfaces), simplified by reducing the triangle patch count to accelerate voxelization and saved as .stl file format. Finally, they are imported into MATLAB R2015b for the rest of the abstraction. Scanned parts can occasionally have small artifacts as seen in Fig. 3a However, those artifacts are insignificant, and rarely need to be explicitly removed.

1) Scaling: The scanned playdough model is a smaller mock-up of the desired structure where the user has to specify the size of the desired final structure. In other words, the user is giving only the shape information with the playdough and the final size of the desired object is another parameter. There could be a hard-coded constant gain to scale up the 3D model to get the real-life object size. However, making an exactly scaled model means more effort for the user since removing or adding material to change the scale of the sculpture may not be simple. Additionally, the user may want to change the size of the object after some use. In such scenarios, having an adjustable parameter κ eases the user's role. In this paper, κ is assumed to be the maximum



(a) (b)

Fig. 3. (a)Initial view of the 3D scan and (b) after pre-processing when the mesh is ready for the voxelization.

length of the object along any x-y-z axes and it can be set on the fly. We envision that the user can use the GUI or a physical object such as knob or slider bar to define the scale. In summary, κ is the only user parameter other than the playdough shape in the proposed interface. Fig. 3b shows the scaled model with $\kappa = 420mm$.

2) Orientation: Orientation of the mesh is critical for the processing step of the proposed interface since the Roombots modules will be placed along the orthogonal axes as explained in Sec. II-D. In a stationary 3D scanning system, the orientation of the coordinate frame of playdough structure would be fixed with respect to the world coordinate frame and the user could simply rotate the playdough sculpture to input the desired orientation. Although this could be a nice feature, our orientation information is not reliable due to the handheld 3D scanner. A workaround is to rotate the mesh manually. Alternatively, we propose an auto rotate method in the pre-processing step.

We assume that the user is working on a flat surface, e.g. table, with the surface normal pointing in reverse gravity direction. As a result, the only unknown orientation axes that needs to be optimized is rotation with respect to the surface normal. The problem can be formulated as

$$\underset{\theta \in [0^\circ, 90^\circ)}{\arg\max}\ f(\theta, \mathcal{M}) \tag{1}$$

where f is the occupancy function, θ is the rotation with respect to the surface normal where the sculpture is placed and $\mathcal{M}$ denotes the scanned and scaled mesh.

C. Voxelization of the 3D model

The power of the shape representation of a voxel space highly depends on the voxel resolution. When the resolution is low, the voxel space suffers from aliasing. Voxelizing the $\mathcal{M}$ with Roombots-size voxels is likely to result in drastic shape deformations. Hence, we firstly voxelize the $\mathcal{M}$ with relatively high resolution to minimize the severe effects of aliasing and obtain the high resolution voxel set $\mathcal{V}_h$. The voxelization for $\mathcal{V}_h$ is based on the generic ray intersection method similar to the one that is described by [23]. The whole space is discretized uniformly (center of each cell space representing a voxel area) and voxels confined in the $\mathcal{M}$ are assigned into $\mathcal{V}_h$. The voxels of $\mathcal{V}_h$ obtained by voxelizing the mesh in Fig. 3b are plotted in Fig. 4a.

Secondly, we create a uniform low resolution voxel set $\mathcal{V}_l$ on the same space. Voxels in $\mathcal{V}_l$ are large and have the size of half of a Roombots module. Unlike $\mathcal{V}_h$, where all voxels are binary, $\mathcal{V}_l$ associates an occupancy value to each voxel. That is one of the main breaking points where the proposed solution goes from a generic to a specific application for Roombots. In our implementation, we set the size of large voxels of $\mathcal{V}_l$ (they will be called v_l) to be five times larger than the small voxels of $\mathcal{V}_h$ (they will be called v_s). That ratio between the voxel sizes will be called ρ. So, the space covered by a half Roombots module is represented by 5^3 v_s.

1) Grid offset selection: The accuracy of shape representation capability with large voxel set $\mathcal{V}_l$ relies on the discrete offset (o) value between $\mathcal{V}_l$ and $\mathcal{V}_h$. The offset is the discrete position translation (in 3D space) of $\mathcal{V}_h$ on top of $\mathcal{V}_l$ with the step size of v_s edge length. The occupancy value of a single voxel in ($\mathcal{V}_l$) is defined as

$$\mathcal{V}_l(x_i, y_i, z_i) := \sum_{x_i-\lfloor\rho/2\rfloor}^{x_i+\lfloor\rho/2\rfloor} \sum_{y_i-\lfloor\rho/2\rfloor}^{y_i+\lfloor\rho/2\rfloor} \sum_{z_i-\lfloor\rho/2\rfloor}^{z_i+\lfloor\rho/2\rfloor} \frac{\mathcal{V}_h(x, y, z)}{\rho^3} \quad (2)$$

which corresponds to total number of v_s in the space defined by the given v_l divided by total number of v_s that could fit in the same space. When $\mathcal{V}_l$ is shifted by the size of one v_s in any x-y-z axis, the occupancy of voxels in $\mathcal{V}_l$ changes. Before proceeding further, we define another set $\mathcal{V}_{lo}$ which a subset of $\mathcal{V}_l$ that consists of elements of $\mathcal{V}_l$ that have minimum 0.5 occupancy.

For better shape representation of $\mathcal{V}_l$, its voxels should be occupied as densely as possible. In more rigorous terms, the optimization problem can be formulated as

$$\underset{p\in[0,\rho-1)}{\arg\max}\, g(p, \mathcal{V}_{lo}, \mathcal{V}_h, \rho) \quad (3)$$

where $p \in \mathbb{N}$ and the aim is finding the optimal position offset $p = [p_x, p_y, p_z]$ for $\mathcal{V}_h$ that maximizes the fitness function g which is defined as

$$g(p, \mathcal{V}_{lo}(p), \mathcal{V}_h, \rho) := \frac{[2\sum \mathcal{V}_{lo}(p) - \mathbf{card}(\mathcal{V}_{lo}(p))] * \rho^3}{\mathbf{card}(\mathcal{V}_h)} \quad (4)$$

In Eq. 4, the first term in the numerator (when multiplied by ρ^3) denotes the number of occupied v_s in the set $\mathcal{V}_{lo}$ (i.e. occupied space) and the second term comes from the number of extra voxels that could fit in $\mathcal{V}_{lo}$ (i.e. empty space). Thus, the fitness function is rewarding the higher occupancy, whereas it is penalizing low occupancy. Finally, the term is normalized with the total number of voxels in $\mathcal{V}_h$. Finding the optimal offset between $\mathcal{V}_h$ and $\mathcal{V}_l$ is automated and the implementation is given in Alg. 1.

For better clarification we will revisit the auto-rotate method. The computational implementation of the rotation procedure, is given in Alg. 2. The implementation of Eq. 1 is a brute force method in which the resolution of θ is taken to be 5°.

2) Search space selection (reduction): The following subsections will be explaining the construction/search part of the interface. Having a small search space usually reduces the computation time of the search algorithm. Hence we select only $\mathcal{V}_{lo}$ to be filled with Roombots modules. The resulting $\mathcal{V}_{lo}$ of the mesh shown in Fig. 3b can be seen in Fig. 4b.

Algorithm 1 Find the offset yielding the highest occupancy

Require: Voxel set $\mathcal{V}_h$ obtained
procedure FINDBESTOFFSET($\mathcal{V}_h, \rho$)
  $o_{max} \leftarrow 0$ ▷ o is occupancy
  $p_{opt} \leftarrow [0, 0, 0]$
  for (each p offset) **do** ▷ p is 3D in xyz
    $\mathcal{V}_{lo} \leftarrow$ CREATEVLO(p,$\mathcal{V}_h$,ρ) ▷ Eq. 2
    $o \leftarrow$ CALCULATEFITNESS(p,$\mathcal{V}_h$,$\mathcal{V}_{lo}$,ρ) ▷ Eq. 4
    if $o > o_{max}$ **then**
      $o_{max} \leftarrow o$
      $p_{opt} \leftarrow p$
  return o_{max}, p_{opt}

Algorithm 2 Rotate mesh to maximize occupancy

Require: Mesh $\mathcal{M}$ is scanned
procedure ROTATEMESH($\mathcal{M}$)
  $\theta_{max} \leftarrow 0$
  $o_{max} \leftarrow 0$
  for (each angle θ) **do**
    $\mathcal{M}_t \leftarrow$ ROTATEZ($\mathcal{M}, \theta$)
    $\mathcal{V}_h \leftarrow$ VOXELIZE($\mathcal{M}_t$) ▷ $\mathcal{V}_h$ is voxel set
    $o, p \leftarrow$ FINDBESTOFFSET($\mathcal{V}_h, \rho$)
    if $o > o_{max}$ **then**
      $o_{max} \leftarrow o$
      $\theta_{max} \leftarrow \theta$
  $\mathcal{M} \leftarrow$ ROTATEZ($\mathcal{M}, \theta_{max}$)
  return $\mathcal{M}$

D. Search Space Definitions and Initial Module Placement

In the rest of the section, the search approach used to fill $\mathcal{V}_{lo}$ with Roombots modules is explained. For a comparative study, two of the well known exhaustive search algorithms, breadth first search (BFS) and depth first search (DFS) are implemented. The discrete search space is $\mathcal{V}_{lo}$ which is a 3D voxel array. Each node is defined as a connected structure that is possible to be constructed with Roombots. Each layer corresponds to total number of modules in the structure. For

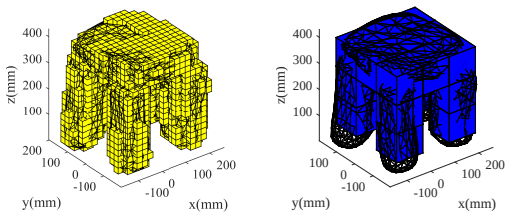


Fig. 4. (a) High resolution voxel space to approximate the 3D model accurately and (b) low resolution voxel space in which the voxel size is equal to a half Roombots module.

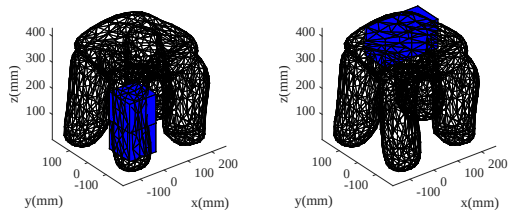


Fig. 5. The initial module can be placed in any location in $\mathcal{V}_{lo}$. Two examples can be seen in (a) and (b)

instance layer zero is the empty $\mathcal{V}_{lo}$ and layer-1 is a only single Roombots module placed in $\mathcal{V}_{lo}$. The goal node (n_G) is the state where either all voxels of $\mathcal{V}_{lo}$ are occupied (when $\mathbf{card}(\mathcal{V}_{lo})$ is an even number) or just a single voxel is left empty (when $\mathbf{card}(\mathcal{V}_{lo})$ is an odd number) assuming the search problem is optimally solvable. The globally optimal solution is the structure which yields the highest possible occupation. By the term optimality, we also refer to set of structures that are completely constructible with Roombots, unlike the case discussed in Sec. II-D. When the $\mathcal{V}_{lo}$ does not lead to an optimal solution, we need to search all possible structures to come up with the best matching structure.

In order to ensure globally optimal solution, the start node (n_S) should be the empty $\mathcal{V}_{lo}$ and in the first iteration all possible initial placements should be inserted into the search queue/stack. Eventually, all initial placements lead to the optimal solutions when $\mathbf{card}(\mathcal{V}_{lo})\%2 = 0$, provided that the initial placement does not result in non-optimal free space (i.e. dividing the free space and each new space blob having odd number of voxels). Further discussion on optimality will be given in Sec. III-F.2. Thus, we pick a random optimal initial seed module and start the construction search with that one. Fig. 5 illustrates two different initial configurations.

E. Construction with Breath First Search

BFS gives a structured baseline to analyze the construction process. It exhaustively searches for the all possible structures and stops only when all alternatives are tested. However, the expansion factor of the search space is very high exponential. A single Roombots module has 10 surfaces. In free space, another module can attach to any of the surfaces in five different ways, resulting in 46 ($50 - 4$ repetitions removed) different possibilities (children nodes) in the second layer. The number of surfaces increases, resulting in even higher number of children per parent node in the 3rd layer. In such an exponential search space, any branch that can be pruned helps to improve the total computation time. Hence, we do the pruning with (i) history keeping and (ii) optimal placement check. The history all of the opened nodes is recorded and we do not push the child in the search queue if it already exists in the history. The history is implemented with *map container* with unique keys generated for each structure to optimize the time spent checking if the node has already been opened. The flow of the procedure can be seen in Alg. 3. Once BFS ends, the user has a selection of possibilities listed in highest occupancy order as shown in Fig. 6.

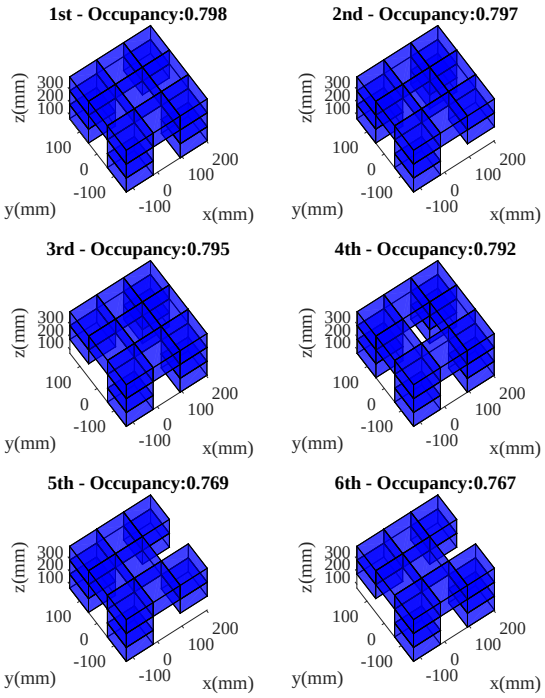


Fig. 6. Six structures resulting from the construction based on BFS with n_s shown in Fig 5a. Structures are ordered in the occupancy order. Note that the $\mathcal{V}_{lo}$ has odd number of Voxels for the desired stool structure which means one voxel remains unoccupied. The user can pick any of the resulting structure. Even though the 1st structure has a higher occupancy, the 4th structure would be a better stool in terms of functionality.

F. Construction with Depth First Search

The structure of the DFS is more suitable for the addressed construction problem particularly when the structure can be built with Roombots and the optimal solution exists. Therefore, we implemented DFS with the same structure of Alg. 3 by only changing the queue to stack. DFS marches towards goal and stops as soon as the goal state is reached. If we let the DFS to search all possibilities, the results would be the same as BFS since they are both exhaustive methods. However, our DFS implementation is intended to find a quick solution for large search space. Fig. 7 gives the resulting structures of the DFS approach.

1) Structures that cannot be built completely: When the structure cannot be built optimally with Roombots, DFS cannot terminate since n_G does never appear and it searches for all possibilities like BFS. While the exhaustive search still continues, the best results can be displayed to the user and user can stop the search anytime when satisfied with one of the offered solutions.

2) Pruning method based on optimal placement: We propose a pruning method for the implemented search algorithm

Algorithm 3 Construction with BFS

Require: $\mathcal{V}_{lo}$ is obtained

1: **procedure** CONSTRUCTBFS($\mathcal{V}_{lo}$)
2: Initialize empty h ▷ h is the history
3: Initialize q with seed module n_S ▷ q is the queue
4: **while** q is not empty **do**
5: $n_p \leftarrow$ POP(q) ▷ parent node
6: $\mathcal{N}_c \leftarrow$ GETCHILDREN(n_p)
7: **for** (each n_c in $\mathcal{N}_c$) **do**
8: **if** n_c is not in h and optimal **then**
9: PUSH(n_c,q) ▷ child into q
10: PUSH(n_c,h) ▷ child into h
11: n_G = FINDHIGHESTOCCUANCY(h)
12: **return** n_G

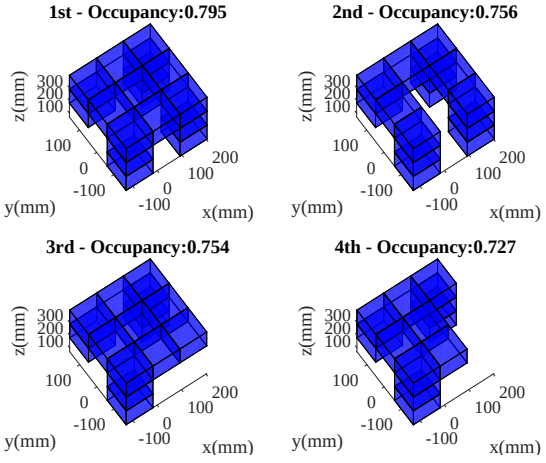


Fig. 7. Construction of modular structures with DFS.

that is mentioned as *optimality check* on the eight line of Alg. 3. If the $\mathcal{V}_{lo}$ can be fully filled with Roombots, we can immediately decide if an opened child will result in non-optimal solution. If the child is dividing the remaining free space into more then one 3D blobs, all blobs should have an even number of voxels. Otherwise, we can prune that branch by discarding the child.

G. User Feedback

One crucial observation in Fig. 7 is that DFS does not suggest a nice functional stool. The main reason is that $\mathcal{V}_{lo}$ has an odd number of voxels and one voxel need to get discarded. However, the displayed results are only for one scan (snapshot) of the playdough. In an interactive continuous case, assuming that the 3D-scanning can be done much faster, the user can still guide the interface towards a desired structure (see Sec.III-G). In order to demonstrate the role of the user feedback, then 3D model is modified to emulate what the user could do. Applying the modification, this time the resulting $\mathcal{V}_{lo}$ has an even number of voxels as seen in Fig. 8 and a functional stool is quickly found by the search method.

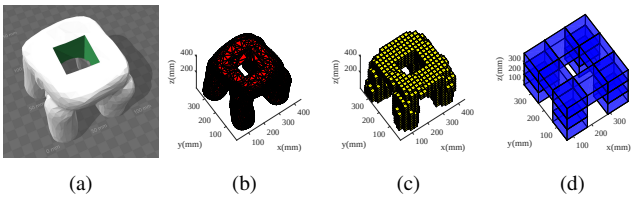


Fig. 8. (a) Creating an artificial feedback by modifying the 3D model directly and running the same procedures on the modified model. Resulting (b) mesh, (c) high resolution voxel space and (d) resulting structure.

TABLE I
COMPARATIVE PERFORMANCE OF THE SUGGESTED METHODS

Object	**# mod.**	**# h_d**	**t_d (s)**	**$t_{d,np}$ (s)**	**t_b (s)**
Stairs wood (h)	7	15	0.10	0.06	0.19
Stool (Fig. 8)	8	16	0.10	0.14	0.27
Fancy chair (d)	10	38	0.14	0.10	17.9
Armchair 1 (a)	11	58	0.16	19.6	85.4
Stairs LEGO (e)	14	66	0.19	105	1171
Sofa 2 (g)	16	108	0.24	NA	NA
Armchair 2 (b)	17	187	0.32	0.24	NA
Bookshelf (c)	20	387	2.70	545	NA
Sofa 1 (f)	25	329	1.91	NA	NA

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

In order to test the performance of the computational part of the proposed interface, we created the test set seen in the Fig. 9 (a)-(h). The common property of all of those models is that they have an even number of v_l and they can be filled optimally after scaling them to Roombots size. For the scaling of the models, we have considered smaller mock-up furnitures rather than the human size because we only have 13 Roombots modules in Biorobotics Laboratory and we focused more on the physical experiments we can do in the future. Fig. 9 (i)-(p) shows the resulting Roombots structures of each model in the test set. The number of modules needed for each model can be seen in Tab. I. The same table also reports the computation time for each model when processed with the BFS, DFS and DFS with no pruning check as well as number of different structures tested in history (h_d) until finding the solution with DFS. Computations are done on a PC with an Intel i7-6700HQ CPU and 16GB RAM.

The Tab. I suggests that the proposed pruning method enables the search to be used in real-time applications in the Roombots project. The BFS time clearly shows the explosive effect in computation time since the search space is highly exponential with respect to the size of the structure. The *NA* marks in the table represents that not all possible structures could be tested in 30 minutes. To put this into perspective: when the *Sofa 1* object is searched for approximately 7 hours, roughly 12 million structures are tested and BFS was still in the 11th layer (11 modules placed out of 25 module model).

A. Effect of Initial Module Placement

The location of initial module also effects optimality. For instance when the stool example given in Fig. 6 is initialized with the seed module given in Fig. 5b, two results having

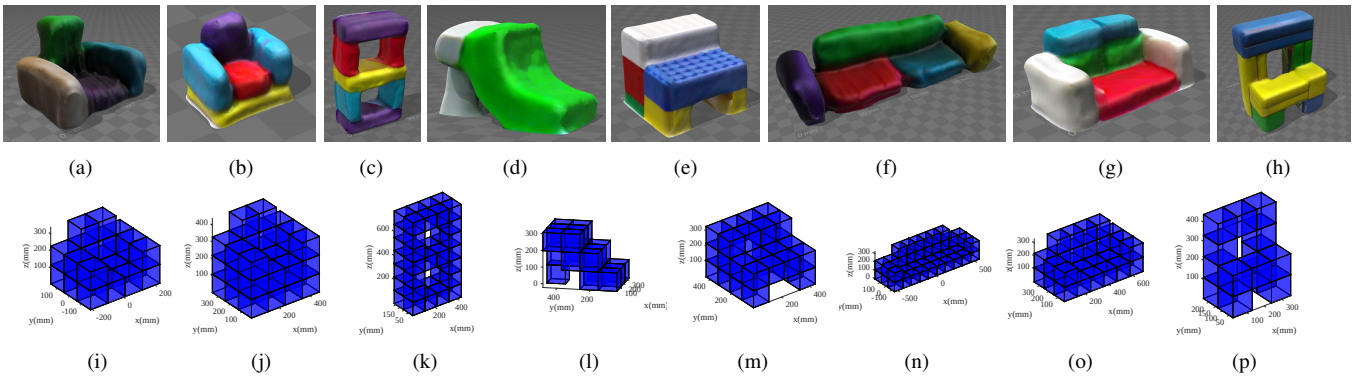


Fig. 9. Different objects used for benchmarking and the corresponding found Roombots structures.

highest occupancy can be seen in 10. Note that the second option was not available previously, due to the placement of the seed module. However, DFS already returns the first solution and global optimality of the solution relies on the user interaction where user should iterate the playdough sculpture to achieve the desired structure.

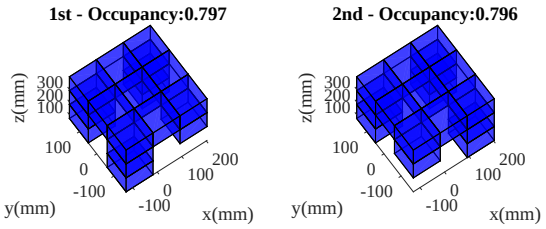


Fig. 10. Best two results of BFS when the seed module is taken as shown in Fig. 5b.

B. Different construction materials

Modeling clay is not the only option for our interface. LEGO®, wood pieces, any deformable or brick-like material can be a tangible medium in this work. The test set already has models made out of LEGO® and wood pieces. One selection criteria can be the connection capabilities. For instance, LEGO® blocks or wood pieces can only be stacked on top of each other, but, Roombots modules has docking surfaces also on the sides. In that regard, playdough can represent capabilities of Roombots better.

Even active building blocks such as Active Cubes could have been used. For instance, [24] estimates CAD models from Voxel model made with robotic blocks. It is almost the reverse of our problem and could simplify our interface. There would be no need for the (i) scanning, (ii) initial voxelization, (iii) rotation search and (iv) large voxel offset search steps.

C. Hardware demonstrations

We have manually replicated some of the test set models with Roombots and report the *Stool* in Fig. 11. We also tested the reverse process by 3D scanning Roombots structure, as seen in Fig 11c, and running the algorithm to replicate (or potentially by changing size scale) the Roombots structure with another set of Roombot. In that case, active Roombots blocks are replacing playdough to control yet another set of Roombots (or even another SRMR).

Fig. 11. The stool example from (a) playdough to (b) Roombots and further (c) reverse scanning the Roombots structure.

D. Limitations of the proposed solution

Our demonstrated system has three major limitations: (i) 3D scanning is not real time, thus we only demonstrate the proof of concept, (ii) it does not use the full potential of Roombots and assumes them as uniform construction bricks, (yet we can still demonstrate a large subset of the full capabilities) and (iii) scaling up the size of desired structures leads to exponential computation time.

The suggested DFS with history and the pruning check can solve a reasonably sized search space in real time which is sufficient for us since we have only 13 modules. However, scaling up to more than 40-module-structures starts suffering from computation time and getting a quick solution depends on the chance. For large free spaces, additional strategies are needed.

V. CONCLUSIONS AND FUTURE WORK

In this paper we are proposing a novel, tangible human robot interaction to control the shape of SRMRs. The proposed interface consists of multiple computation steps. The highest computation load is resulting from the search to construct the given shape optimally. The attempted solution space is highly exponential and pruning branches as soon as possible plays a crucial role to have real-time performing interface.

The proposed solution can be applied to any lattice type SRMRs. Having a SRMR with the shape of a single voxel eliminates the construction step. Thus, large scale scenarios can be be built very fast. If a SRMR has a more complex shape, the rule to get new child nodes in the construction step should be updated according to the new shape. The rest of the implementation is generic.

Properties inherent to TUIs also apply to this work. For example, a visually disabled person can control Roombots using the TUI. He may not make use of the suggested visual feedback, but, can always touch and feel the final structure after self-reconfiguration and modify the playdough when needed.

The future work in this line of research will focus on incorporating inverse kinematics into the proposed interface to be able to build more complex structures. Real-time 3D scanning is an enabling factor for this work. Hence, we will adopt an automated real time 3D scanning technique. The final work plan is including passive components to replace some of the robotic modules to the search algorithm which would reduce the cost of real systems and possibly boost upscaling properties of the interface.

ACKNOWLEDGMENT

Authors would like to thank to Ayşe Nur Sunal and Olguta Robu for creating Play-Doh structures and providing alternative construction materials. Authors also would like to thank to M.J. BRUNNER Inc. for granting the permission to use Fig. 2.

REFERENCES

[1] A. Sproewitz, R. Moeckel, M. Vespignani, S. Bonardi, and A. J. Ijspeert, "Roombots: A hardware perspective on 3d self-reconfiguration and locomotion with a homogeneous modular robot," *Robotics and Autonomous Systems*, vol. 62, no. 7, pp. 1016–1033, July 2014.

[2] M. Yim, W. m. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics Automation Magazine*, vol. 14, no. 1, pp. 43–52, March 2007.

[3] S. Bonardi, J. Blatter, J. Fink, R. Moeckel, P. Jermann, P. Dillenbourg, and A. J. Ijspeert, "Design and evaluation of a graphical iPad application for arranging adaptive furniture," in *2012 IEEE RO-MAN*, Sept. 2012, pp. 290–297.

[4] A. Oezgur, S. Bonardi, M. Vespignani, R. Moeckel, and A. J. Ijspeert, "Natural user interface for Roombots," in *2014 RO-MAN: The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, Aug. 2014, pp. 12–17.

[5] M. Mutlu, S. Bonardi, M. Vespignani, S. Hauser, A. Bernardino, and A. J. Ijspeert, "Natural user interface for lighting control: Case study on desktop lighting using modular robots," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Aug. 2016, pp. 288–293.

[6] S. Bonardi, R. Moeckel, A. Sproewitz, M. Vespignani, and A. J. Ijspeert, "Locomotion through Reconfiguration based on Motor Primitives for Roombots Self-Reconfigurable Modular Robots," in *Proceedings of ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 2012, pp. 1–6.

[7] H. Ishii, D. Lakatos, L. Bonanni, and J.-B. Labrune, "Radical Atoms: Beyond Tangible Bits, Toward Transformable Materials," *interactions*, vol. 19, no. 1, pp. 38–51, Jan. 2012.

[8] K. Nakagaki, L. Vink, J. Counts, D. Windham, D. Leithinger, S. Follmer, and H. Ishii, "Materiable: Rendering Dynamic Material Properties in Response to Direct Physical Touch with Shape Changing Interfaces," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 2764–2772.

[9] D. Anderson, J. L. Frankel, J. Marks, A. Agarwala, P. Beardsley, J. Hodgins, D. Leigh, K. Ryall, E. Sullivan, and J. S. Yedidia, "Tangible Interaction + Graphical Interpretation: A New Approach to 3d Modeling," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 393–402.

[10] R. Watanabe, Y. Itoh, M. Asai, Y. Kitamura, F. Kishino, and H. Kikuchi, "The Soul of ActiveCube: Implementing a Flexible, Multimodal, Three-dimensional Spatial Tangible Interface," *Comput. Entertain.*, vol. 2, no. 4, pp. 15–15, Oct. 2004.

[11] H. S. Raffle, A. J. Parkes, and H. Ishii, "Topobo: A Constructive Assembly System with Kinetic Memory," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '04. New York, NY, USA: ACM, 2004, pp. 647–654.

[12] B. Piper, C. Ratti, and H. Ishii, "Illuminating Clay: A 3-D Tangible Interface for Landscape Analysis," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '02. New York, NY, USA: ACM, 2002, pp. 355–362.

[13] H. Ishii, C. Ratti, B. Piper, Y. Wang, A. Biderman, and E. Ben-Joseph, "Bringing Clay and Sand into Digital Design Continuous Tangible user Interfaces," *BT Technology Journal*, vol. 22, no. 4, pp. 287–299, Oct. 2004.

[14] S. Johnson and A. P. Thomas, "Squishy Circuits: A Tangible Medium for Electronics Education," in *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '10. New York, NY, USA: ACM, 2010, pp. 4099–4104.

[15] W. Gao, Y. Zhang, D. C. Nazzetta, K. Ramani, and R. J. Cipra, "RevoMaker: Enabling Multi-directional and Functionally-embedded 3d Printing Using a Rotational Cuboidal Platform," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ser. UIST '15. New York, NY, USA: ACM, 2015, pp. 437–446.

[16] K. Stoy, "Using cellular automata and gradients to control self-reconfiguration," *Robotics and Autonomous Systems*, vol. 54, no. 2, pp. 135–141, Feb. 2006.

[17] K. Stoy and R. Nagpal, "Self-repair through scale independent self-reconfiguration," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 2, Sept 2004, pp. 2062–2067 vol.2.

[18] J. W. Romanishin, K. Gilpin, and D. Rus, "M-blocks: Momentum-driven, magnetic modular robots," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 4288–4295.

[19] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3d model acquisition," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 438–446, July 2002.

[20] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '11. New York, NY, USA: ACM, 2011, pp. 559–568.

[21] S. Follmer and H. Ishii, "KidCAD: Digitally Remixing Toys Through Tangible Tools," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 2401–2410.

[22] J. Straub, B. Kading, A. Mohammad, and S. Kerlin, "Characterization of a large, low-cost 3d scanner," *Technologies*, vol. 3, no. 1, pp. 19–36, 2015.

[23] S. Patil and B. Ravi, "Voxel-based representation, display and thickness analysis of intricate shapes," in *Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05)*, Dec 2005, pp. 6 pp.–.

[24] H. Ichida, Y. Itoh, Y. Kitamura, and F. Kishino, "Interactive Retrieval of 3d Shape Models Using Physical Objects," in *Proceedings of the 12th Annual ACM International Conference on Multimedia*, ser. MULTIMEDIA '04. New York, NY, USA: ACM, 2004, pp. 692–699.